LUISS

Degree Program in Management and Computer Science

Chair Data Analysis for Business

Markov Graph-Based Approaches for Recommendation Systems: An Application to the Movie Domain

Margherita Bernardini - 284051

CANDIDATE

Prof. Alessia Caponera

SUPERVISOR

Academic Year 2024/2025

Ai miei genitori, per
avermi sempre supportata e
per avermi resa la persona
che sono.

Contents

Introduction and Overview

The ongoing usage of online platform in daily life, from entertainment (e.g., Spotify, Netflix) to shopping (e.g., Amazon Shopping, E-bay), has brought the need to understand and predict user preferences. As recommendation systems have become essential to digital applications, the ability to model, analyze, and anticipate user behavior is crucial for improving user experience and engagement, but also for increasing company profitability.
This thesis explores the theoretical foundations and practical applications of graphical models, specifically Markov graphical models, for modeling dynamic user preferences. The work is divided into three main parts:

We begin by introducing Markov graphical models, focusing on both directed and undirected graphs represents relationships between variables. The chapter provides an overview of Bayesian networks, Markov chains, and Markov networks, establishing the mathematical and statistical computations necessary for sequential modeling.

Following, we present the article "Where To Next? A Dynamic Model of User Preferences", with a focus on the Preference Transition Model, a dynamic framework designed to model how users' genre preferences evolve over time. The PTM is analyzed in detail, including its notation, statistical computation, and parameter estimation. We also discuss the connections between the PTM Model, Baysian Networks and Markov Chains.

In the final section, we implement a model ispired by the PTM on the MovieLens 25M dataset, a large-scale collection of user movie ratings. The main objective of this application is to investigate the dynamis of genre transition among users. The results are visualized and interpreted, providing insight into genre prediction and the underlying processes of recommendation systems.

This thesis aims to offer a comprehensive study of user preference dynamics. The methods and analysis presented can be applied to other domains where the understanding of sequential preferences is crucial.

Chapter 1

Markov Graphical Models

1.1 Introduction on graphical models

A graph G is the composition of a set of nodes, better known as vertices V, with a set of edges which joins some pairs of vertices. In the context of graphical models, each vertex represents a random variable, providing a visual representation of the joint distribution over a set of variables. There exist two main types of graphical models, namely the undirected and the directed one. For the sake of this discussion, we will focus on directed graphical models, including **Markov Chains** and **Bayesian networks**, which are particularly relevant for the modeling of user preferences over time. However, we will also include a brief overview of undirected graphical models, better known as **Markov random fields** or **Markov networks**.

1.2 Directed Graphical Models - Bayesian Networks

A **directed graph** [10] consists in a set of vertices V (or nodes) connected by directed edges, or arrows. These graphs provide a helpful and structured way to represent independence relations between random variables. Each vertex in the graph corresponds to a variable, and the arrow that connects node X to node Y indicates that X is a direct parent of Y in the dependency structure. Directed graphs are commonly used to represent probabilistic relationships and to facilitate factorization of the joint distribution.
A well-known form of directed graphical model is the **Directed Acyclic Graph (DAG)**, a class that avoid directed cycles, meaning that it is not possible to start at a node and follow a sequence of arrows that leads back to the starting node.
Directed graphs are also referred to as **Bayesian networks** when related with probability distribution, even though the term might be misleading. While DAGs are compatible with Bayesian inference, they are not actually Bayesian. Frequentist and Bayesian methods can also be used for estimation and inference; therefore, it is better to use the term *directed graphical model* to define them.

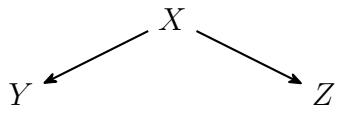

Figure 1.1: A simple directed graph with vertices $V = \{X, Y, Z\}$ and edges $E = \{(X, Y), (X, Z)\}$.

1.2.1 Conditional Independence

The main concept regarding directed graphical models is **conditional independence**. It allows us to divide complex joint distributions into simpler and more understandable conditional distributions.

Let X, Y, and Z be random variables. We say that X and Y are **conditionally independent** given Z, denoted $X \perp Y \mid Z$, if, for all values of x, y, and z:

$$f_{X,Y|Z}(x, y|z) = f_{X|Z}(x|z) f_{Y|Z}(y|z) \tag{1.1}$$

This means that, once Z is known, knowing Y provides no additional information about X. Equivalently:

$$f(x|y,z) = f(x|z) \tag{1.2}$$

Conditional independence satisfies some key properties:

Theorem 1 (Properties of Conditional Independence)**.** *Let X, Y, Z, W be some random variables, and let $U = h(X)$ be a function of X. Then:*

- $X \amalg Y|Z \implies Y \amalg X|Z$ *(Symmetry)*
- $X \amalg Y|Z \quad$ *and* $\quad U = h(X) \implies U \amalg Y|Z$ *(Invariance under functions)*
- $X \amalg Y|Z \quad$ *and* $\quad U = h(X) \implies X \amalg Y|(Z,U)$ *(Augmentation)*
- $X \amalg Y|Z \quad$ *and* $\quad X \amalg W|(Y,Z) \implies X \amalg (W,Y)|Z$ *(Contraction)*
- $X \amalg Y|Z \quad$ *and* $\quad X \amalg Z|Y \implies X \amalg (Y,Z)$ *(Intersection)*

Note that, except for the first four properties, the Intersection property requires all the events to have positive probabilities.

1.2.2 Directed Acyclic Graph (DAG)

A **directed graph** $G = (V, E)$ consists of a set of vertices V and a set of directed edges $E \subset V \times V$. Each node corresponds to a random variable, and each directed edge $(X, Y) \in E$ indicates a directed relationship from X to Y (graphically speaking, there is an arrow going from X to Y), and we refer to X as the **parent** of Y and to Y as the **child** of X. The set of all parents of a node X is denoted by π_X or $\pi(X)$.

- Two variables X and Y are said to be **adjacent** if an arrow connects them in either direction.
- A **directed path** between two variables X, Y is a sequence of directed edges pointing in the same direction connecting one variable to the next, such as $\boldsymbol{X \to \cdots \to Y}$.
- If the direction of the edges of a sequence of adjacent vertices are ignored, the sequence is referred to as an **undirected path**. For instance, $\boldsymbol{Z \leftarrow X \rightarrow Y}$.
- We say that X is an **ancestor** of Y if there exists a directed path from X to Y (or if $X = Y$). Conversely, Y is a **descendant** of X.

Colliders and Non-colliders

A sequence of three nodes along a path, with the structure $\boldsymbol{X \to Y \leftarrow Z}$ is known as a **collider** at Y. Intuitively, this occurs when two arrows converge into the same node.
Any configuration along a path that is not a collider, for example of the form $\boldsymbol{X \to Y \to Z}$ or $\boldsymbol{X \leftarrow Y \to Z}$, is known as a **non-collider** at Y.
Note that a collider and a non-collider status is *path-dependent*, meaning that a node may be a collider on one path and a non-collider on another. For example, in a graph where the sequence $\{X, Y, Z\}$ forms a collider, Y may still be a non-collider along a different path such as $\{X, Y, W\}$.
If the parents of a collider node are not adjacent (i.e., they are not connected by an edge), the collider is called an **unshielded collider**.

Cycles and Acyclicity

We define a **cycle** a directed path that begins and ends at the same vertex. A graph that does not contain any cycles is called a **acyclic**.
In this case, we say that the graph is a **Directed Acyclic Graph (DAG)**. Acyclicity is an essential property in probabilistic graphical models, as it ensures a well-defined factorization of the joint distribution and avoids infinite recursive dependencies.

Probability and DAGs

We define a **Directed Acyclic Graphs (DAGs)** all those graphs which provide a visual representation of casual relationships given a set of different variables. In particular, each arrow indicates a casual relationship that is either true or assumed to be.

Let G be a Directed Acyclic Graph with vertices $V = (X_1, ..., X_n)$. If we suppose that P is a distribution over V, with probability function f, then we can say that **P is Markov to G**, or equivalently that G is the representation of P, if

$$f(v) = \prod_{i=1}^{k} f(x_i|\pi_i) \tag{1.3}$$

with π_i indicating the set of parents of X_i, in the graph. Note that $M(G)$ is the notation that represents the set of all distributions of G. Graphically speaking, we can represent a DAG as follows:

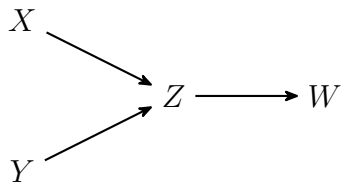

Figure 1.2: A simple DAG.

Theorem 2. *A distribution $P \in M(G)$, if and only if, for every variable W, the following **Markov Condition** holds:*

$$W \perp W_D = \pi_W, \tag{1.4}$$

where

- *W_D indicates the set of all the variables in the graph with the exemption of its parents and descendants,*
- *π_W denotes the set of parents of W.*

In other words, the theorem states that, if the condition holds, then the variable is independent of its *nonrelevant past*, once the set of parents π_W is known.

Independence Relations

Thanks to the Markov Condition it is possible to derive other independence relations suggested by a directed acyclic graph. In general, any of these independencies may,consequently imply some other oblivious relations.
For instance, if we consider the following DAG:

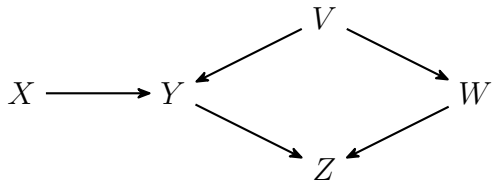

Figure 1.3: Example of a DAG

From the Markov condition, we obtain the following independence relations:

- $V \perp X$

- $X \perp \{V, W\}$
- $Y \perp W \mid \{V, X\}$
- $W \perp \{X, Y\} \mid V$
- $Z \perp \{V, X\} \mid \{Y, W\}$

Moreover, the relations above also imply the following condition:

$$\{W, Z\} \perp X \mid \{V, Y\}, \tag{1.5}$$

To find such an independence relation, we use the **d-separation** (which stands for *directed separation*) method, a criterion that identifies independence in DAGs.

Rules of d-separation
If we consider the DAGs in Figure1.4 and Figure1.5, we have the following rules:

1. If Y **is not** a collider in the path between X and Z, then:
   - X and Z are **d-connected** without conditioning on Y,
   - X and Z are **d-separated** given Y.
2. If Y **is** a collider in the path between X and Z, then:
   - X and Z are **d-separated** without conditioning on Y,
   - X and Z are **d-connected** given Y.
3. Conditioning on a descendant of a collider (Figure1.5) is the same thing as conditioning on the collider itself (e.g. X and Z are **d-separated**, but, given the descendant W, they are **d-connected**).

Figure 1.4: Four examples of DAGs with the bottom-right one being a **collider**.

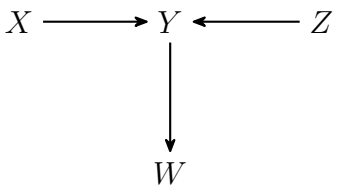

Figure 1.5: A collider with a **descendant** (W).

Definition 1. *Let X and Y be distinctive vertices in a DAG, and let W be a set of vertices that does not contain X and Y. Then, X and Y are **d-separated** given W if there is no undirected path U between them such that:*

- *every collider in U has a descendant in W, and*
- *a non collider on U belongs to W.*

Of course, this also extends to sets of vertices:

Theorem 3. *Let A, B and C be disjoint sets of vertices. Then*

$$A \perp B \mid C \tag{1.6}$$

if and only if A and B are d-separated by C.

For example, if we consider three disjoint sets A, B and C, with $A, B \neq 0$, then, by the definition above, the two sets A **and** B **are d-separated given** C, if, for every vertex $X \in A$ and $Y \in B$, both vertices are d-separated given C. On the contrary, the sets that are not d-separated are d-connected.

Different graphs may sometimes denote the same independence relations. For example, let us take a DAG G and denote by $I(G)$ the set of all independence relations denoted by G. Then, two DAGs G_1 and G_2, defined by the some set of variables W, are **Markov Equivalent** if $I(G_1) = I(G_2)$, that is, if they imply the same relations.

Theorem 4. *Given two DAGs G_1 and G_2, they are said to be **Markov Equivalent** if and only if*

1. *their skeletons are identical, **skeleton**$(G_1) =$ **skeleton**(G_2), and*
2. *they have the same set of unshielded colliders.*

For any DAG G, the **skeleton** of G, is the *undirected* graph obtained by removing all directions from the edges of the graph.

Estimation of DAGs

There exist some ways to estimate Directed Acyclic Graphs (DAGs). In particular, assuming that we are given a DAG G and a set of data $X = \{V_1, ..., V_n\}$ generated from a distribution f consistent with GG, we want to know how to estimate the distribution function f.
A common approach is to assume a parametric form for each conditional density. That is, for each variable X, we want to model

$$f(x \mid \pi_x; \theta_x), \tag{1.7}$$

where,

- π_x denotes the set of parents of the variable X in the graph, and
- θ_x denotes the parameters of the conditional distribution.

The likelihood formula is given by:

$$L(\theta) = \prod_{i=1}^{n} f(V_i \mid \theta) = \prod_{i=1}^{n} \prod_{j=1}^{m} f(X_{ij} \mid \pi_j, \theta_j) \tag{1.8}$$

where,

- X_{ij} is the value of variable X_j at the i-th observation, and
- θ_j denotes the parameters associated with the conditional distribution of X_j

The parameters then are to be estimated by calculating the maximum likelihood.

1.3 Markov Chains

Stochastic processes describe the evolution of random variables over time. Two of the most crucial examples include the **Bernoulli process**, which models a sequence of independent binary trials with two possible outcomes, success or failure, such as coin tosses, and the **Poisson process**, which models the occurrence of random events over time, widely used to understand and predict different scenarios.
However, for the purpose of this research, the focus will be on **discrete-time Markov chains**, which extend these ideas by allowing the system to move among multiple states according to transition probabilities that depend only on the current state. This property makes **Markov chains** a useful tool for analyzing and modeling the dynamics of user preferences over time.

1.3.1 Discrete-Time Markov Chains

Discrete-time Markov chains [1][6] are a sequence of random variables where the system evolves in discrete time steps, indexed by an integer n, at each step of which the chain has a **state** X_n that belongs to a finite set S called the **state space**. We assume the state space to be

$$S = \{1, 2, ..., m\} \tag{1.9}$$

for a set of *positive* integers m.
The evolution of the chain is determined by its transitioned probabilities p_{ij}; In particular, if the current state $i \in S$, the probability that the next state is $j \in S$, is determined by

$$p_{ij} = \mathbb{P}(X_{n+1} = j \mid X_n = i), \tag{1.10}$$

with the following conditions to be satisfied:

- $p_{ij} \geq 0 \; \forall \; ij \in S$
- $\sum_{j=1}^{m} p_{ij} = 1 \; \forall \; i \in S$

The key assumption of the **Markov Property** is that the transition probabilities p_{ij} depend only on the current state i no matter its history. Mathematically speaking,

$$\mathbb{P}(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, ..., X_0 = i_0) = \mathbb{P}(X_{n+1} = j \mid X_n = i) = p_{ij} \tag{1.11}$$

for all,

- n times,
- $i, j \in S$ states, and
- $i_0, ..., i_{n-1}$ possible sequences of earlier stages

Therefore, as Equation 1.11 demonstrates, the conditional distribution of the next state X_{n+1}, depends only on the current state X_n. In the case when the probabilities $p_{ii} > 0$, the chain can remain in the same state even in the next step. When the case occurs, we are talking about **self-transition**.
The transition probabilities p_{ij} of a Markov chain are usually arranged into a transition matrix as follows:

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mm} \end{pmatrix}. \tag{1.12}$$

whose entries in the ith rows and jth colums are the probabilities p_{ij}.
Another representation of the model is with the transition graph, where the nodes define the states and the edges (alternatively called arcs when curved) define the possible transition between them. Such representation makes the performance of the chain more intuitive, by also highlighting its structure.

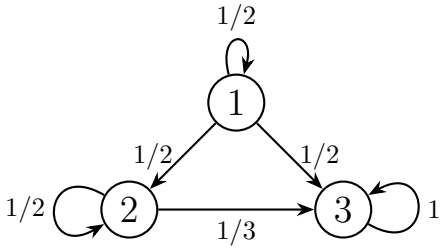

Figure 1.6: Example of a Markov chain

Estimation: probability of a sequence of States

A Markov chain also allow us to compute the probability of a sequence of state, which directly follows the lead of the **Multiplication Rule** in probability.
For any sequence $(i_1, i_2, ..., i_n)$ we have:

$$\mathbb{P}(X_0 = i_0, X_1 = i_1, ..., X_n = i_n) = \mathbb{P}(X_0 = i_0)p_{i_0 i_1}p_{i_1 i_2}...p_{i_{n-1} i_n} \tag{1.13}$$

If the is known and it is equal to i_0, the probability of observing the states' sequence $(i_1, i_2, ..., i_n)$ is

$$\mathbb{P}(X_1 = i_1, ..., X_n = i_n \mid X_0 = i_0) = p_{i_0 i_1}p_{i_1 i_2}...p_{i_{n-1} i_n} \tag{1.14}$$

Graphically speaking, the state sequence corresponds to a path of arcs of the transition graph, and the probability such path is the product of the probabilities of each arc along the path.

Proof. Equation 1.13 above can be verified using the Markov property, as follows:

$$\mathbb{P}(X_0 = i_0, X_1 = i_1, ..., X_n = i_n) = \mathbb{P}(X_n = i_n \mid X_0 = i_0, ..., X_{n-1} = i_{n-1}) \cdot \mathbb{P}(X_0 = i_0, ..., X_{n-1} = i_{n-1}), \tag{1.15}$$

since

$$\mathbb{P}(X_n = i_n \mid X_0 = i_0, ..., X_{n-1} = i_{n-1}) = p_{i_{n-1} i_n}, \tag{1.16}$$

by recursively repeating this argument backwards, we can reach the result on Equation 1.13. □

n-Steps Transition Probabilities

In many applications of Markov chains, we are interested in predicting the probability distribution of the state at some **future time** given the current state. This concept will later be applied to the case studied in this thesis. This application is described by the **n-step transition probabilities**, defined as

$$r_{ij}(n) = \mathbb{P}(X_n = j \mid X_0 = i), \tag{1.17}$$

where r_{ij} is the **probability** that, starting from state i, the chain will be in state j, after n steps. These probabilities can be computed recursively using the **ChapmanKolmogorov equation**:

$$r_{ij}(n) = \sum_{k=1}^{m} r_{ik}(n-1)p_{kj}, \ with \ n > 1 \tag{1.18}$$

with initial condition $r_{ij}(1) = p_{ij}$.

Proof. Equation 1.18 above can be verified by applying the total probability formula. Conditioning on the intermediate state at time $n-1$, we obtain

$$P(X_n = j \mid X_0 = i) = \sum_{k=1}^{m} \mathbb{P}(X_{n-1} = k \mid X_0 = i) \cdot \mathbb{P}(X_n = j \mid X_{n-1} = k). \tag{1.19}$$

where, the variable k denotes the intermediate state at time $n-1$, and, the sum runs over all possible states in the state scope. By the Markov property, the last term depends only on the current state k, which justifies the expression above. □

To make it more intuitive and to understand its behavior, it is convenient to arrange the values $r_{ij}(n)$ into a matrix $R(n)$, called the **n-step transition probability matrix**. The entry in the i-th row and j-th column gives the probability of moving from state i to state j in exactly n steps.

1.3.2 Classification of State

Over long times, the behavior of the n-step transition probability matrix $R(n)$ can exhibit different asymptotic patterns, such as convergence to steady-state probabilities or absorption into special states. Some states, once visited, may even be revisited. This section classifies and defines states according to the long-term frequency with which they are visited.

Accessibility and Recurrent State

Definition 2. *A State j is said to be **accessible** from a state i if there exists a number of steps $n \geq 1$ such that the probability of transitioning from i to j in n steps is positive: $r_{ij}^{(n)} > 0$.*

Equivalently, there exists a sequence of states $(i, i_1, i_2, ..., i_{n-1}, j)$ connecting i to j, where each transition occurs with positive probability.

Let $A(i)$ denote the se of states *accessible* from i;

Definition 3. *A state i is said to be **recurrent** if it is possible to return to i from every state that is accessible from it.*

This means that, starting from a recurrent state, it is always possible to return to it, and, eventually, it will be revisited infinitely often, over time.

Contrary to the definition above,

Definition 4. *A state is **transient** if it is not recurrent. For a transient state i, there exists at least one $j \in A(i)$ such that i cannot be reached again.*

Therefore, there is a positive probability that a state i can never be reached again, meaning that, transient states are visited only a finite number of times.
It is important to know that the classification as recurrent or transient depends only on the structure of the transition graph, and not on the specific values of the transition probabilities.

Recurrent Classes and Markov Chain Decomposition

The set of states accessible from a recurrent state i forms a **recurrent class**, meaning that all states within the set are mutually accessible, and no other state outside the class is accessible from them. This concept is expressed as follows:

$$A(i) = A(j) \; forall \; j \in A(i) \tag{1.20}$$

Definition 5. *A Markov chain can be decomposed into one or more recurrent classes, along with possible transient states.*

The Markov chain decomposition satisfies the following conditions:

- States in a recurrent class are mutually accessible but cannot be reached from other recurrent classes,
- Transient states are not accessible from any recurrent state, and
- Every transient state eventually leads to at least one recurrent state.

Decomposition provides a useful framework to analyze the chain's behavior both in a long-term or short-term perspective as, if the chain starts in a recurrent class, it remains there indefinitely, visiting all states infinitely often, and if the chain starts in a transient state, the trajectory consists of an initial transient segment followed by a recurrent class segment.

Periodicity

A recurrent class can also be characterized by periodicity.

Definition 6. *A recurrent class is **periodic** if its states can be partitioned into $d > 1$ disjoint subsets $S_1, S_2, ..., S_d$ such that transitions from any state in S_k lead only to states in S_{k+1}, with transitions froms_d returning to S_1.*

In this case, the chain revisits each subset only after d steps.

Otherwise, if no such partition exists, the class is considered **aperiodic**. In an aperiodic class, however, it is possible, after some number of steps, to reach any state in the class from any other. By checking these conditions, it is possible to easily verify whether a class is periodic or not.

1.3.3 Steady-State Behavior

In a Markov chain with a single recurrent, aperiodic class, the n-step probabilities $r_{ij}^{(n)}$ of transitioning from one state to another, converge as $n \to \infty$ to fixed values that are independent from the starting state. These probabilities, denoted by π_j, are known as **steady-state probabilities**, satisfy the following balance equations:

$$\pi_j = \sum_{k=1}^{m} \pi_k p_{kj}, \qquad \sum_{j=1}^{m} \pi_j = 1, \tag{1.21}$$

and they represent the long-term likelihood of occupying state j. Transient states have $\pi_j = 0$, while recurrent states have $\pi_j > 0$. It is important to know that for steady-state analysis, the chain should be aperiodic, as, if the class happens to be periodic, the steady-state probabilities would not exist.

Long-Term Frequency Interpretation.

The steady-state probabilities π_j can be interpreted as the long-term expected fraction of time the chain spends in each state j over a long period of time. More precisely, for a Markov chain with a single aperiodic recurrent class, we have:

$$\pi_j = \lim_{n\to\infty} \frac{v_{ij}(n)}{n}, \tag{1.22}$$

where $v_{ij}(n)$ is the expected number of visits to state j in the first n transitions, starting from state i.
Similarly, the long-term frequency of transitions from state j to state k is given by:

$$\lim_{n\to\infty} \frac{q_{jk}(n)}{n} = \pi_j p_{jk}, \tag{1.23}$$

where $q_{jk}(n)$ is the expected number of times the chain moves from j to k in n steps.
This interpretation provides an intuitive explanation for the balance Equation1.21: the fraction of time the chain spends in state j is equal to the sum of the fractions of transitions into j from all other states, and it is expressed as follows,

$$\pi_j = \sum_{k=1}^{m} \pi_k p_{kj}. \tag{1.24}$$

1.4 Undirected Graphical Models - Markov Networks

In an **undirected graph** [5] G, the edges have no direction. The edges are assigned values which represent the strength of conditional dependence between the corresponding random variables. Consequently, the absence of an edge between two vertices indicates that the corresponding random variables are conditionally independent, given the other variables.

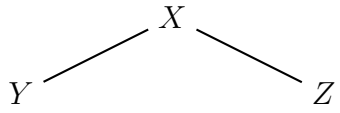

Figure 1.7: An example of an undirected graph with vertices $V = \{X, Y, Z\}$ and set of edges $E = \{(X, Y), (X, Z)\}$.

A graph G is defined as a pair (V, E), where V is the set of vertices and E is the set of edges. Considering the stated definition, some properties can be described:

- Two vertices X and Y are considered *adjacent* if they share an edge, and it is denoted by $X \sim Y$,

- A *path* $X_1, X_2, ..., X_n$ is a sequence of vertices where each pair (X_{i-1}, X_i) is connected by an edge, with $i = 2, .., n$,
- A *complete graph* is one in which every pair of vertices is connected by an edge,
- A *sub-graph* $U \subseteq G$ consist of a subset of vertices together with the respective connecting edges.

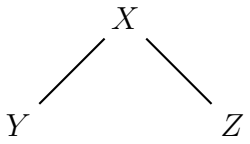

Figure 1.8: The pair of vertices $\{X, Y\}$ and $\{X, Z\}$ are adjacent, notably $X \sim Y$ and $X \sim Z$

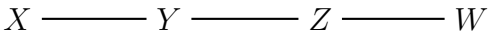

Figure 1.9: Example of a path

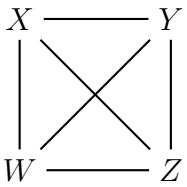

Figure 1.10: Example of a complete graph.

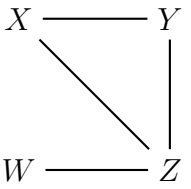

Figure 1.11: The graph G_1, with vertices $U = \{X, Y, Z\}$ is a subgraph of the graph G, with vertices $V = \{X, Y, Z, W\}$.

1.4.1 Pairwise Markov Independencies and Global Markov Properties

As stated previously, in a Markov graph G, the absence of an edge between two vertices indicates a relationship of conditional independence between the corresponding random variables. This statement is expressed with the notation $X \perp Y|rest$, where '*rest*' is referred to all the other vertices, and they are known as the **pairwise Markov Independencies** of G.

In a graph G, we call a *separator* the set of vertices which disconnect the remaining sub-graph, if removed. Taking a subgraph composed of the nodes A, B and C, then it is said that C **separate** A and B if every path that connects them passes through at least one node in C. A key property of **separators**, is that they divide the graph into **conditionally independent components**. Specifically, the definition is expressed with the notation $A \perp B|C$, if C separates A and B, and these are known as the **global Markov Properties** of G. It is important to note that the pairwise and global Markov properties are equivalent (meaning that any distribution which satisfies one property also satisfies the other), for graphs with positive distributions. The global Markov property makes it possible to decompose graphs into smaller sections, which highly simplifies their understanding and implementation. It is then essential to define the concept of clique. A **clique** is a complete subgraph, meaning that, it is a set of vertices all adjacent to each other. A clique is called **maximal** if, no other vertices can be added to it, and still yield a clique. We call a **potential** any positive function.

The equivalence between the pairwise and global Markov properties, allows the deduction of **global conditional independencies** by inspecting a graph's structure (e.g., missing edges). Furthermore, regarding positive distributions, the **Hammersley-Clifford theorem** guarantees the factorization of the joint probability as a product of non-negative potential functions over the maximal cliques of the graph:

$$f(x) = \frac{1}{Z} \prod_{C \in \rfloor} \psi_C(x_C) \tag{1.25}$$

Here, Z is a constant known as the partition function, and ψ_C are what is known as clique potentials, positive functions that encode the dependencies between the variables in each clique.

This is particularly useful for designing inference algorithms, such as the *junction tree algorithm*, for the computation of marginal and low order probabilities.

It is worth noting that, while undirected graphs express **conditional independence**, they do not determine the order or strength of higher-order dependencies. For instance, a complete graph with more than three nodes may represent a distribution that includes either only pairwise or full three-way interactions. For sake of simplicity, the majority of models focus on pairwise Markov graphs, where only second-order interactions (edges) are used. These models create a balance between representational ability and computational efficiency.

Chapter 2

"Where To Next? A Dynamic Model of User Preferences"

2.1 Introduction

In recent years, understanding the dynamics of user preferences has become increasingly important for the development of effective recommendation systems. The article "*Where To Next? A Dynamic Model of User Preferences*" [8] introduces an interesting approach to modeling how user behavior evolves, using a dynamic statistical approach to capture how preferences shift over time. For the sake of my research, I am using this study as a methodological guide, aiming to reproduce and adapt its core ideas in a different setting. While the original study focused on location-based data, this project applies the same modeling principles to the **MovieLens 25M** dataset, a dataset which was also referenced in the original paper, to explore changes in user preferences over time in the domain of movie recommendations. The main goal is to see how users interests in movie genres change across a defined period (particularly in 5 years), and to build a system that can predict such changes.

2.2 Preference Transition Model (PTM)

The Preference Transition model is a dynamic model programmed to capture how users' preferences evolve over time.

2.2.1 Notation and problem Statement

We denote with M the set of users which interacts with items over time. We associate each item to one of N classes, and we divide the period into T time windows. We define the *class* as domain-dependent. In the context of this thesis, we use the **MovieLens 25M** dataset as our study domain. In this framework, users interact with movies, and each movie is associated with one or more tags. For the sake of simplicity and modeling, we will consider a five-year interval, from 2015 to 2019, included, and we will treat each tag as a mutually exclusive item class and assign each movie to a single representative tag (e.g., the most common one).

Let us denote $n_{ij}^t \in \mathbb{N} \cup 0$ the number of interactions a user i had with an item (movie) belonging to class j (tag) during the time window t. Note that such interactions could coincide with events like watching or rating a movie. The **vector of interaction counts** for user i in time windows t is defined as:

$$n_i^t = [n_{i1}^t, n_{i2}^t, \ldots, n_{iN}^t]. \tag{2.1}$$

Each entry in $\boldsymbol{n_i^t}$ corresponds to the number of movies from class j watched or rated by user i during period t.
The **activity of the user i** during the time window t, denoted by $\boldsymbol{\xi_i^t}$, is defined as the sum of all interactions across genres:

$$\xi_i^t = \sum_{j=1}^{N} n_{ij}^t. \tag{2.2}$$

This value represents the total number of movies the user interacted with in the designated time period. We assume that each user has at least 1 activity in every time window, so, $\xi_i^t > 0$ for all $i \in \{1, \ldots, M\}$ and $t \in \{1, \ldots, T\}$.

The **class distribution vector $\boldsymbol{\pi_i^t}$** identifies the relative preference of user i for each genre in time window t. It is defined with the vector:

$$\pi_i^t = [\pi_{i1}^t, \pi_{i2}^t, \ldots, \pi_{iN}^t] = \frac{n_i^t}{\pi_i^t}. \tag{2.3}$$

Each component π_{ij}^t represents the proportion of interactions the user had with genre j at time window t. Note that, by construction, the components of π_i^t sum up to 1:

$$\sum_{j=1}^{N} \pi_{ij}^t = 1. \tag{2.4}$$

The main prediction task is to forecast each user's future interaction vector, based on their past behavior. Particularly, given the set of past interaction vectors:

$$\{n_i^t\}, \quad \text{for all} \quad i \in \{1, \ldots, M\}, t \in \{1, \ldots, T\} \tag{2.5}$$

we aim to predict the netx interaction vector $\boldsymbol{n_i^{T+1}}$ for each user i:

$$\hat{\boldsymbol{n}}_{\boldsymbol{i}}^{\boldsymbol{T+1}} = [n_{i1}^{T+1}, n_{i2}^{T+1}, \ldots, n_{iN}^{T+1}]. \tag{2.6}$$

This vector gives us an estimate of how many items (that is, movies) from each genre the user will interact with, in future time windows *T+1*.

2.2.2 Statistical Model

To implement the evolution of preferences over time, we introduce a dynamic probabilistic framework called the **Preference Transition Model (PTM)**. Our goal is to predict the future genres distribution a user will interact with in a following time window, based on their prior behavior. In particular, the model will allow us to study the progress of n_i^t over time and therefore the distribution π_i^t, establishing the existing relationships between the different classes. The model is divided in two parts:

- Using an Exponentially weighted moving average (**EWMA**) for temporally modeling preferences;
- Using Poisson and Multinomial distributions for predicting future activity and genre distribution.

Temporal Preference Evolution via EWMA

We start by specifying how users' preferences evolve through time using a recursive update rule. We define the distribution $\tilde{\pi}_i^t$ for user i in time window t, which takes into account a user's preference history, adding more weight to recent interactions. This is computed with the Exponential Weighted Moving Average (EWMA):

$$\tilde{\boldsymbol{\pi}}_{\boldsymbol{i}}^{\boldsymbol{t}} = (1-\gamma)\tilde{\pi}_i^{t-1} + \pi_i^t, \quad \text{with} \quad \tilde{\pi}_i^1 = \pi_i^1 \tag{2.7}$$

In this case, $\gamma \in [0,1]$ is an *exploration* parameter that measures how fast the model interacts with different classes. We assume that γ is the same for all users. A higher γ puts more weight on recent behavior while a smaller γ focus on longer-term memory.

Poisson and Multinomial distribution

The model is then used to predict future behavior focusing on both the magnitude of activity and the genre structure of preferences.
Firstly, the total number of interactions a user i will have in a time window $t+1$, denoted by ξ_i^{t+1}, is computed using the Poisson distribution;

$$\boldsymbol{\xi}_{\boldsymbol{i}}^{\boldsymbol{t+1}} \sim \text{Poisson}(\xi_i^t) \tag{2.8}$$

This indicates that a user i activity level a *t+1* tends to stay around the one in the previous time window t, but it varies in a stochastic way.
Secondly, given the predicted activity $\boldsymbol{\xi}_{\boldsymbol{i}}^{\boldsymbol{t+1}}$, the genre classification of their interactions is sampled with a Multinomial distribution, using a reweighted version of the distribution $\tilde{\pi}_i^t$;

$$n_i^{t+1} \quad | \quad \xi_i^{t+1}, \tilde{\boldsymbol{\pi}}_{\boldsymbol{i}}^{\boldsymbol{t}} \sim \text{Multinomial}(\xi_i^{t+1}, \tilde{\pi}_i^t \boldsymbol{A}) \tag{2.9}$$

In this case, $A = \{a_{jk}\} \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$ is a stochastic matrix that captures the transition probabilities between genres, where:

- Each row of A sum up to 1: $\sum_{k=1}^{N} a_{jk} = 1, \quad \text{for all} \quad j;;$
- Each entry $\boldsymbol{a_{jk}}$ indicates the probability that the preference for genre j, a time t, will lead to an interaction with genre k, at time $t+1$;
- All the entries are non-negative: $a_{jk} \geq 0 \quad \forall \quad j,k = 1,...,N.$

It is important to know that the activity (1.8) and the class distribution (1.9) are computed *separately*. This provides more interpretability as each component is to be analyzed and predicted independently. Furthermore, the model formulation has a strong resemblance to a **first-order Markov chain**, where the **states** of the chain correspond to the genres (classes) in the dataset. This can be clearly observed under the assumption that $\gamma = 1$, where no historical activity is considered and, consequently, the current distribution π_i^t fully displays the user's current preference at time t:

$$a_{jk} = \mathbb{P}(\text{genre } k \text{ at time } t+1 \quad | \quad \text{genre } j \text{ a time } t) \tag{2.10}$$

Here, a_{jk} defines the probability that a user moves from genre j to genre k from one time window to the next. Using this equation (1.10), the matrix A assumes a probabilistic structures, which enables important insights about the relations between the classes. For instance, if a user mainly watches *Action* films at time t, and at time $t+1$ they watch *Adventure* films with high probability, then

the transition probability $a_{\text{Action,Adventure}}$ will be relatively large, indicating a strong genre shift pattern between these two classes. It is important to know that the matrix A is not constrained to be symmetric. That is,

$$a_{jk} \neq a_{jk} \tag{2.11}$$

The assymmetry introduces the notion of directionality between classes of items.

2.2.3 Parameter Estimation

The Preference Transition Model relies on two key parameters, the exploration parameter γ and the class transition matrix A. Jointly estimating these two parameters is significantly difficult. Specifically, any change in the value of γ needs a further computation of all EWMA distributions $\tilde{\pi}_i^t$ each user i and time window t, making the optimization process inconvenient. One possible way to overcome this could be by optimizing a joint objective function over both parameters (γ, A), for instance, via coordinate descent. However, even with γ fixed, estimating A alone is computationally expensive. For this case, the authors propose a two-step estimation procedure: firstly we estimate γ, then, given γ, we estimate A conditionally.

Estimation of γ

To estimate the exploration parameter (γ, A), we begin by simplifying the model. The idea is to temporarily assume that there are no transitions between the classes, meaning that, the transition matrix A is equal to the identity matrix I_N. Under such assumption, the goal is to select (γ, A) so that the EWMA distribution $\tilde{\pi}_i^t$ best approximates the next distribution $\tilde{\pi}_i^{t+1}$. This can be done by minimizing the **negative log-likelihood (NLL)** of the observed counts n_{ij}^{t+1} given the EWMA distribution:

$$\gamma = \frac{1}{M} \sum_{i=1}^{M} \left\{ \arg \min_{\gamma_i \in [0,1]} \left[- \sum_{t=1}^{T-1} \sum_{j=1}^{N} n_{ij}^{t+1} \log \tilde{\pi}_{ij}^{t}(\gamma_i) \right] \right\} \tag{2.12}$$

Note that:

- M is the number of users;
- N is the number of genre (or classes);
- T is the number of time windows;
- n_{ij}^{t+1} is the number of times user i interacted with genre j at time $t+1$;
- $\tilde{\pi}_{ij}^{t}(\gamma_i)$ is the EWMA distribution for user i and genre j at time t, depending on γ.

This loss function penalizes inconsistencies between the predicted preference distribution $\tilde{\pi}_i^t$ and the actual user activity at the next step $t+1$. The parameter (γ, A) is individually optimized for each user and then averaged over all users to obtain a global value. To prevent computational issues caused by the limit case $log(0)$, the EWMA distribution at time $t = 1$ is initialized as follow:

$$\tilde{\pi}_i^1 = \frac{n_{ij}^1 + \frac{1}{N}}{\xi_i^1 + 1} \tag{2.13}$$

In a Bayesian framework, this initialization corresponds to the mean of a Dirichlet posterior distribution for π_i^1, assuming a conjugate Dirichlet prior with all parameters set to $\frac{1}{N}$.

Estimation of A conditional on γ

Once (γ, A) is determined, we can proceed to estimate the class transition matrix A, which represents the evolution of preferences from one class to another over time. The matrix is estimated by minimizing the negative log-likelihood of the observed class counts n_i^{t+1}. This leads to the optimization problem:

$$\hat{A} = \arg\min_A \left\{ -\sum_{t=1}^{T-1} \sum_{i=1}^{M} \sum_{j=1}^{N} n_{ij}^{t+1} \log \left(\sum_{k=1}^{N} \tilde{\pi}_{ik}^t a_{kj} \right) \right\} \tag{2.14}$$

Here:

- $\tilde{\pi}_{ik}^t$ is the weighted preference of user i for genre k at time t, which incorporates both previous and current interactions.
- a_{kj} is the probability of switching from genre k at time t to genre j at time $t+1$;

The inner summation $\sum_{k=1}^{N} \tilde{\pi}_{ik}^t a_{kj}$ computes the overall probability that a user i will prefer genre j at time $t+1$, based on all previous preferences of genre k. Afterwards, this is used in the likelihood function, weighted by the actual number of interaction with genre j by user i at time $t+1$.
This optimization problem is subject to the fact that $A = [a_{jk}]$ must be a row-stochastic matrix. Particularly:

- $a_{jk} > 0 \quad \text{for all} \quad j, k;$
- $\sum_{k=1}^{N} a_{kj} = 1 \quad \text{for all} \quad j.$

The computational complexity of evaluating the objective function (1.14) is:

$$\mathcal{O}\left(N \sum_{t=1}^{T} \sum_{i=1}^{M} \sum_{j=1}^{N} (n_{ij}^t > 0) \right) \tag{2.15}$$

This is quite efficient when the counts n_{ij} are sparse (which can be often the case in a large-scale datasets like the MovieLens 25M that will be used for our project). To solve this constrained optimization, the **Adam algorithm** is used. The Adam algorithm is a stochastic optimization technique based on adaptive moment estimation, which provides convergence properties even in high-dimensional, sparse settings.

2.2.4 Prediction

The purpose of the Preference Transition Model (PTM) is to predict a users future preferences and activity levels. In particular, after estimating the model parameters γ and A, we aim to predict:

- the corresponding **class distribution** π_i^{t+1}, that represents the users normalized preference vector at time $t+1$.
- the **expected** number of **interactions** n_i^{t+1} a user i will have with each class at time $t+1$;

Prediction of the class distribution

Given the estimated transition matrix A, that models how preferences switch between genres over time, and the smoothed distribution $\tilde{\pi}_i^t$, which outlines the users past behaviors, to predict the expected distribution of preferences at time $t+1$, denoted $\mathbb{E}[\tilde{\pi}_i^{t+1}]$, we can approximate it using the conditional expectation. This results in:

$$\hat{\pi}_i^{t+1} = \tilde{\pi}_i^t A \tag{2.16}$$

Note:

- $\tilde{\pi}_i^t \in \mathbb{R}^N$ is the smoothed class distribution at time t, computed using EWMA (1.7);
- $A \in \mathbb{R}^{N \times N}$ is the transition matrix where entry a_{kj} represents the probability of moving from class k at time t to class j at time $t+1$;
- The new resulting prediction $\hat{\pi}_i^{t+1}$ gives us the expected distribution over genres (or classes) at time $t+1$.

Prediction of interaction counts

We now proceed to predict the **expected** number of **interactions** $\mathbb{E}[\boldsymbol{n_i^{t+1}}]$ a user will make in each class at time $t+1$. We know from equation (1.8) that the users **activity level** ξ_i^{t+1} follows a Poisson distribution with mean ξ_i^t. Conditional on ξ_i^{t+1}, the number of **interactions across genres** (or classes) n_i^{t+1} is implemented as a multinomial distribution (1.9).

To compute the expected interaction counts, we use the **law of iterated expectation (LIE)**, which states that *"the expected value of* $\mathbf{X}$ *is equal to the expectation over the conditional expectation of* $\mathbf{X}$ *given* $\mathbf{Y}$*. More simply, the mean of* $\mathbf{X}$ *is equal to a weighted mean of conditional means"*. That is:

$$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[\mathbb{X}|\mathbb{Y}]] \tag{2.17}$$

By applying the law above to our problem we get:

$$\mathbb{E}[n_i^{t+1}] = \mathbb{E}[\mathbb{E}[n_i^{t+1}|\xi_i^{t+1}, \tilde{\pi}_i^t]] = \mathbb{E}[\xi_i^{t+1}] \times \tilde{\pi}_i^t A \tag{2.18}$$

Since by the Poisson distribution (1.8) we have that $\mathbb{E}[\xi_i^{t+1}] = \xi_i^t$, the final prediction for the interactions in each class is:

$$\hat{n}_i^{t+1} = \xi_i^t \times \tilde{\pi}_i^t A \tag{2.19}$$

The equation gives us a predicted count vector $\hat{n}_i^{t+1} \in \mathbb{R}^N$ which represents how many interactions user i is expected to make with items from each genre at time $t+1$.

2.3 Relation with Markov Graphical Models

The Preference Transition Model (PTM) is closely related to the Markov graphical models theory. In particular, with the concepts of the Markov chains, which aims in predicting the probability distribution of the state at some future time given the current state. Similarly, the PTM Model describes user preference evolution over time, but as a stochastic process governed by a transition mechanism.
In particular,

- In a **Markov chain**, the state of a system at time $t+1$ depends only on its current state t, through a transition matrix P, where each row specifies the probabilities of moving from one state to the others.
- In the **PTM Model**, the normalized preference vector $\pi_{t,i}$ for the user i plays the role of a state distribution. The transitio matrix A in the PTM describes the evolution of the weights assigned to different preference categories (e.g., genres) from one time window to the next. Thus, the equation

$$\pi_{t+1,i} \approx \pi_{t,i} A \tag{2.20}$$

mirrors the Markov chain recursion, except that here the state is not a single category, but a probability distribution over categories.

From a graphical model perspective, a Markov graph (DAG or undirected) encodes conditional dependencies among variables. The PTM applies the same idea to user preferences by assuming a first-order Markov property: the distribution of future preferences depends only on current preferences, not directly on the past.
In this sense, the PTM can be seen as an extension of a Markov chain from discrete states to probability distributions over item classes. Instead of moving between single states, as in a classical Markov chain, the PTM model transitions between distributions of preferences across categories (such as movie genres).

Chapter 3

Model Application - Movielens Project

3.0.1 Introduction

The main goal of this project is to design and apply a model inspired by the PTM Model in the article Where To Next? A Dynamic Model of User Preferences [8], which is able to study the' behavior by identifying how references shift over time. As previously stated, the original study focused on a data set in the music domain (a Spotify internal data set). In this project, we apply the model to the **MovieLens 25M** dataset[3], a dataset also referenced in the original paper, to explore changes in user preferences over time in the domain of movie recommendations.

3.0.2 MovieLens 25M Dataset

The Movielens Dataset[4] express peoples expressed preferences for movies at a particular time. These data were taken from the MovieLens website, since 1998; a system which asks its user to register their rating of a movie in order to receive a personalized movie recommendation. For the project, we used two of the six datasets included in the folder, namely movies and ratings.
The dataset, *movies.csv*, is comprised of the following features:

- **movieId**: A unique identifier for each movie.
- **title**: The title of the associated movie, which also include the year of release in parentheses.
- **genres**: List of genres that characterize the movie, selected from selected from the following selection: Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western

The dataset, *ratings.csv*, is comprised of the following features:

- **userId**: A unique identifier for each user (starting from 1).
- **movieId**: A unique identifier for each movie.
- **rating**: Float, which represents the rating of one movie by one user. Ratings are made on a 5 star scale, with half-star increments (0.5 - 5.0 stars).
- **timestamp**: Indicate when a user rated the associated movie. It is represented in seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

The two datasets are linked through the variable **moviedId**, which serves as a unique identifier for each movie. This key allows us to merge the informations from the ratings file with the corresponding metadata from the movies file.

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

Figure 3.1: First five lines of the movies.csv dataset.

	userId	movieId	rating	timestamp
0	1	296	5.0	1147880044
1	1	306	3.5	1147868817
2	1	307	5.0	1147868828
3	1	665	5.0	1147878820
4	1	899	3.5	1147868510

Figure 3.2: First five lines of the ratings.csv dataset.

3.0.3 Data Preprocessing

Before applying the Model, the two datasets required a series of preprocessing steps to ensure the data were properly formatted, and suitable for temporal analysis. The goal of these steps was to extract, for each user, the sequence of movie interactions over time, with a clear mapping to genres and a uniform timestamp format.

Firstly, a **timestamp conversion** was needed. The dataset provides timestamps in a format that represented the number of seconds since January 1, 1970. These timestamps were converted into a standard datetime format, allowing for easier manipulation and filtering by year.

```
ratings['timestamp'] = pd.to_datetime(ratings['timestamp'], unit='s')
start_date = pd.Timestamp('2015-01-01')
end_date = pd.Timestamp('2019-12-31')
ratings_filtered = ratings[(ratings['timestamp'] >= start_date) & (ratings['
    timestamp'] <= end_date)].copy()
```

The **data** were then **filtered by year**. Since the analysis focuses on user behavior from 2015 to 2019, only ratings within this period were retained. This filtering ensures that the model captures recent user preferences and their evolution within a defined temporal window.

```
movie_ids = ratings_filtered['movieId'].unique()
movies_filtered = movies[movies['movieId'].isin(movie_ids)].copy()
```

Finally, the two datasets were **merged** into a single one. This merged dataset contains, for each rating, the corresponding user ID, movie ID, timestamp, and the genre(s) associated with the movie. The merging step is crucial for simplifying computations and directly associating user activity with genre information.

```
ratings_with_genres = pd.merge(ratings_filtered, movies_filtered, on='movieId', how
    ='left')
```

This resulting dataset, namely *ratings_with_genres*, shown in Figure 3.3 provides a clear overview of the final structure, with all the needed data aligned for future analysis.

	userId	movieId	rating	title	genres	year
0	3	1	4.0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	2015
1	3	29	4.5	City of Lost Children, The (Cité des enfants p...	Adventure|Drama|Fantasy|Mystery|Sci-Fi	2017
2	3	32	4.5	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller	2015
3	3	50	5.0	Usual Suspects, The (1995)	Crime|Mystery|Thriller	2015
4	3	111	4.0	Taxi Driver (1976)	Crime|Drama|Thriller	2017

Figure 3.3: Snapshot of the merged dataset.

Table 3.1 summarizes the main characteristics of the dataset after preprocessing. We analyze the behavior of 1,260 users over a period of five years (2015 to 2019), focusing on their transitions between 1135 genre classes. For our research, each **genre block**, such as $Crime|Drama$, is considered a **unique** genre class without splitting into its individual components. This approach allows us to study users preferences for combined genres and better reflects the structure of the data available for modeling transitions.

Type	Quantity	Description
M	1260	Users
T	5	Years (2015 to 2019)
N	1135	Classes

Table 3.1: Summary of dataset dimensions.

3.0.4 Implementation of Interaction Vectors

Before implementing the Preference Transition Model, it was necessary to represent user activity in a structured form that captures both how often user interacts and how these interactions are distributed across item classes. The model considers a set of M users interacting with N classes (in this case, movie genres) over T time windows. Following the methodology in the article *"Where To Next? A Dynamic Model of User Preferences"* [8], we created the three key vectors necessary for the subsequent analysis.

Interaction Counts n_{ij}^t

The **Interaction Counts** vector n_{ij}^t denotes the number of times user i interacted with an item belonging to class j within time window t.

In the context of this project, n_{ij}^t corresponds to the number of movies of genre j rated by user i during the specific time window t. This measure captures the raw frequency of user interactions with each class and forms the basic input for modeling preference dynamics.

```
df_for_n_t_i = ratings_with_genres[
    ratings_with_genres['genres'].notna() &
    (ratings_with_genres['genres'].str.strip() != '') &
    (ratings_with_genres['genres'].str.strip().str.lower() != '(no genres listed)')
].copy()

n_t_i = df_for_n_t_i.groupby(['userId', 'year', 'genres']).size().unstack(
    fill_value=0)
```

Total Activity ξ_i^t

The **Total Activity** vector ξ_i^t describes the total number of interactions of user i across all classes within time window t. It is calculated as the sum of all n_{ij}^t interactions for that user at that time window, as shown in Equation 2.2.

```
xi_t_i = n_t_i.sum(axis=1).rename('total_activity')
```

This quantity reflects the overall level of engagement of the user during a specific period and is used to normalize class-level interactions.

Class Distribution π_i^t

The **Class Distribution** vector π_i^t captures the proportion of user activity across each class. This normalized vector represents an index of preference for each user i at time t, highlighting which genres the user favored relative to their total activity.

```
n_t_i_aligned = n_t_i.reindex(xi_t_i.index, fill_value=0)
pi_t_i = n_t_i_aligned.div(xi_t_i, axis=0)
```

The vector is calculated using the other two vectors in the following way:

$$\pi_i^t == \frac{n_{ij}^t}{\xi_i^t}. \tag{3.1}$$

3.0.5 Model Implementation

The model implemented in this work was directly inspired by the mathematical framework described in the article "Where To Next? A Dynamic Model of User Preferences [8], with the aim of producing a similar adaptation designed to capture and analyze the temporal evolution of user preferences, focusing on transitions between movie genres. The model is built upon the structure of the MovieLens dataset, using the previously defined interaction vectors to estimate the transition dynamics between genres.

EWMA Distribution and Temporal Smoothing Parameter γ

To account for temporal smoothing of user preferences, an Exponentially Weighted Moving Average (EWMA) is computed for each user. The EWMA incorporates both recent and past interactions, providing a smooth distribution of genre preferences over time.

```
def ewma_distribution_per_user(gamma, n_user_t_i, xi_user_t_i, pi_user_t_i, N):
    T_user = n_user_t_i.shape[0]
    pi_tilde = np.zeros(shape=(T_user-1, N), dtype=np.float32)

    #pi_tilde with t=1
    pi_tilde[0, :] = (n_user_t_i[0, :] + 1/N) / (xi_user_t_i[0] + 1)

    #subsequent pi_tilde
    for t in range(1, T_user-1):
        pi_tilde[t, :] = (1 - gamma) * pi_tilde[t-1, :] + gamma * pi_user_t_i[t, :]

    return pi_tilde
```

The smoothing parameter γ, which controls the contribution of recent versus historical preferences, is formulated according to Equation 2.7, which expresses the smoothed preference vector as:

$$\tilde{\boldsymbol{\pi}}_{\boldsymbol{i}}^{\boldsymbol{t}} = (1-\gamma)\tilde{\pi}_i^{t-1} + \pi_i^t \tag{3.2}$$

where π_i^t is the genre distribution for user i at time t, and $\tilde{\pi}_i^t$ is the EWMA smoothed distribution. In practice, for each user, the optimal value of γ is determined by minimizing the negative log-likelihood of observed interactions, as per Equation 2.12.

```
def gamma_min(gamma, n_user_t_i, xi_user_t_i, pi_user_t_i, N):
  pi_tilde_i = ewma_distribution_per_user(gamma, n_user_t_i, xi_user_t_i,
    pi_user_t_i, N)
  n_user_t_i_final = np.delete(n_user_t_i, 0, axis=0)
  multiplication = np.multiply(np.log(pi_tilde_i), n_user_t_i_final)
  gamma_loss = -(np.sum(multiplication))

  return gamma_loss
```

The smoothing parameter is then optimized for each user to best fit their observed behavior. This optimization is conducted using the Scipy minimization routine *minimize_scalar* [9], applied individually to each user's interaction history.

```
from scipy.optimize import minimize_scalar

optimal_gammas = np.zeros(M)  #array to stores opt. gamma for each user

for i in range(M):
    n_user = n_t_i_np[i]        #(T, N)
```

```
    xi_user = xi_t_i_np[i]      #(T,)
    pi_user = pi_t_i_np[i]      #(T, N)
    res = minimize_scalar(gamma_min, bounds=(0.01, 0.99), args=(n_user, xi_user,
    pi_user, N), method='bounded')
    optimal_gammas[i] = res.x
print("Optimal gammas per user:", optimal_gammas)

gamma_mean = np.mean(optimal_gammas)
```

Genre Transition Matrix A

To model transitions between genres, the PTM Model introduces a genre transition matrix A, as formalized in Equation 2.16 of the article:

$$\hat{\pi}_i^{t+1} = \tilde{\pi}_i^t A \tag{3.3}$$

In this project, each row of A represents the probability distribution over the next genres, conditioned on the current genre. The matrix is constrained to be row-stochastic so that each row sums to one, maintaining probabilistic validity.

The main objective is to find the matrix A that best explains the observed user interactions, by minimizing the negative log-likelihood of observed user interactions given the EWMA-distributed preferences, according to Equation 2.14.

```
def a_loss(A, gamma, n_user_t_i, xi_user_t_i, pi_user_t_i, N):
  T_user = n_user_t_i.shape[1]
  M_shape = n_user_t_i.shape[0]
  a_pi_tilde = np.zeros(shape=(M_shape, T_user-1, N), dtype=np.float32)
  pi_tilde_i = ewma_distribution(gamma, n_user_t_i, xi_user_t_i, pi_user_t_i, N)
  n_user_t_i_final = np.delete(n_user_t_i, 0, axis=1)
  for i in range(0, M_shape):
    for j in range(0, T_user-1):
      a_pi_tilde[i, j, :] = np.matmul(pi_tilde_i[i, j, :], A)
  multiplication = np.multiply(np.log(a_pi_tilde), n_user_t_i_final)
  a_loss = -(np.sum(multiplication))

  return a_loss
```

Optimization via Adam Algorithm

The optimization of the matrix A is performed using the Adam algorithm[7], as implemented in PyTorch. The **Adam Algorithm** is a stochastic gradient-based optimizer mostly used in machine learning to update model parameters based on the gradient of a loss function.

The matrix is initially set to the identity matrix and then it is iteratively updated to minimize the loss. After each update, the matrix is rescaled by normalizing each row, in order to ensure that the learned transition probabilities remain valid.

```
A = torch.nn.Parameter(torch.eye(N, requires_grad=True))  #creates a matrix
optimizer = torch.optim.Adam([A], lr=0.01)
num_epochs = 300

for epoch in range(num_epochs):
    optimizer.zero_grad()
    loss = a_loss_torch(A, gamma_mean, n_t_i_torch, xi_t_i_torch, pi_t_i_torch, N)
    loss.backward()
    with torch.no_grad():
        A.data = torch.clamp(A.data, min=1e-8)
        A.data /= A.data.sum(dim=1, keepdim=True)
    optimizer.step()

    if epoch % 50 == 0:
        print(f"Epoch {epoch}, Loss: {loss.item()}")
```

At convergence, the optimized matrix A encodes the estimated probabilities of transitioning between genres. The minimal loss value indicates the model's goodness to fit to the observed data. A lower loss value indicates that the models predicted genre transitions match closely the actual transitions found in the data. While each row of A indicates the likelihood of moving from one genre to another in the subsequent time window.

```
Minimum loss: 834877.5
Matrix A at minimum loss:
 [[0.09201506 0.00758757 0.00754879 ... 0.00811641 0.00791527 0.00768972]
 ...
 [0.00754747 0.00756203 0.00761444 ... 0.00811641 0.00779924 0.09190485]]
```

The box above shows the final loss reached after optimization and a small section of the transition matrix A.

3.0.6 Visualization

The learned genre transition matrix A enables a visual analysis of the most significant genre transitions. To visualize the results given by the matrix A, a Directed Graph [2] was implemented.

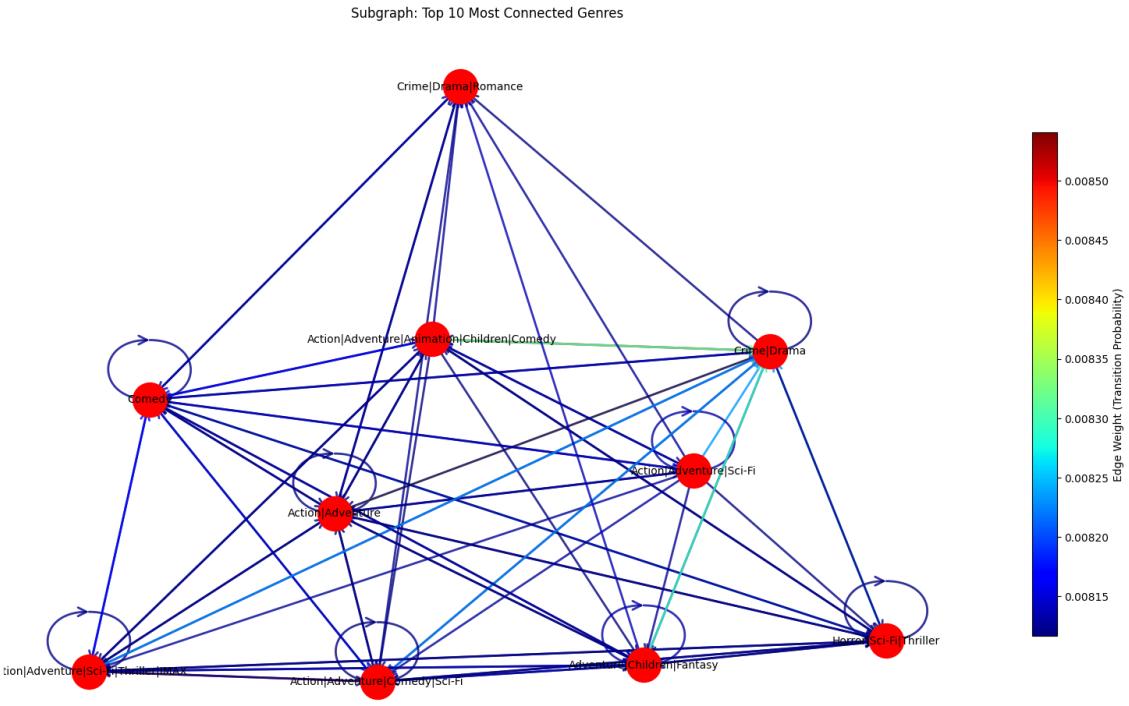

Figure 3.4: Visualization of the top 10 Most Connected Genres.

Figure 3.4 presents a directed graph in which, the nodes represent the genres, and the edges indicates the transition probabilities between genres. Note that the graph is a small subgraph created from the initial transition matrix A.

The edges' color intensity encodes the magnitude of the transition probability, with **warmer colors** indicating a **higher likelihood**, as shown by the color bar. The arrows indicate the direction of transitions, revealing both persistence with self-loops and inter-genre influence.

This visualization provides an intuitive summary of the genre dynamics captured by the model, highlighting which genres are most interconnected based on user behavior.

In particular, the graph is highly interconnected, indicating that users often transition between these top 10 genres. The genre **Comedy** is a central node, with many strong incoming and outgoing transitions. It is possible to see that some genre groups, such as $Crime|Drama$ and $Action|Adventure|Sci-Fi$, also serve as important bridges connecting various other genres. The thickness and color of the edges highlight which transitions are most probable. For example, the brightest edges indicate the strongest genre-to-genre transitions in user behavior.

3.0.7 Model Results

In the article [8], they analyze the relationship between genres in order to better interpret the outcome of the model. Here, we propose these calculations adapted to our model.

Most Likely Next Genre $\hat{j}$

The model provides a transition matrix A, where each element a_{ij} indicates the probability of transitioning from genre i to genre j in the next time step. For each genre i, the **most likely next genre** is the genre $\hat{j}$ corresponding to the maximum value in row i of the matrix:

$$\hat{j} = \arg\max_{j \neq i} a_{ij} \tag{3.4}$$

This analysis provides, for each genre, which other genre, users are most likely to watch next, highlighting the preferred paths users are more likely to follow.
A small section of the output of this analysis is shown as follows:

```
Adventure|Animation|Sci-Fi -> Comedy
Action|Comedy|Drama -> Action|Comedy|Sci-Fi
Comedy|Drama -> Action|Adventure|Animation|Children|Comedy
Action|Sci-Fi|Thriller -> Comedy
Action|Comedy|Horror -> Action|Adventure|Comedy|Sci-Fi
Drama|Romance -> Adventure|Fantasy
Action|Animation|Children|Sci-Fi -> Comedy
```

For example, it is possible to see that users who watch $Adventure|Animation|Sci-Fi$ movies are most likely to next choose $Comedy$ ones, as shown in the first line of output.

Best Connecting Genre $\hat{k}$

The transition matrix A also allows us to investigate indirect transitions between genres. For any pair of genres i and j, the **best connector** genre $\hat{k}$ is the one that maximizes the probability of transitioning from i to j in two steps:

$$\hat{k} = \arg\max_{k \neq i,j} a_{ik} a_{kj} \tag{3.5}$$

This result identifies intermediary genres that most effectively bridge user transitions between different genre types. This is a useful way for understanding the possible multi-step navigation patterns of users and for better designing recommendations.
A small section of the output of this analysis is shown as follows:

```
Adventure|Animation|Children|Comedy|Fantasy -> Action|Adventure|Sci-Fi -> Adventure
    |Drama|Fantasy|Mystery|Sci-Fi
Mystery|Sci-Fi|Thriller -> Comedy -> Crime|Mystery|Thriller
Crime|Drama|Thriller -> Comedy -> Action|Sci-Fi|Thriller
Action|Crime|Sci-Fi -> Action|Adventure -> Drama|War
Action|Adventure|Sci-Fi -> Comedy -> Action|Crime|Drama|Thriller
Comedy|Crime|Drama|Thriller -> Animation|Children|Fantasy|Musical|Romance|IMAX ->
    Crime|Drama
```

The output above shows examples of paths in the form **Start Genre** $\rightarrow$ **Connector Genre** $\rightarrow$ **Target Genre**, showing the intermediate genres as shown in the first row where the genre $Action|Adventure|Sci-Fi$ stands as bridge between the other two genres.

Most Asymmetric Genre Relationships

Another interesting feature that can be identified is the asymmetric relationships between genres. Since the transition matrix A is not necessarily symmetric, it is possible for the probability of moving from genre i to genre j to differ from the probability of transitioning from j to i. The **asymmetry** d_{ij} is computed by calculating the absolute difference for each pair of genres:

$$d_{ij} = |a_{ij} - a_{ji}| \tag{3.6}$$

where pairs with the highest d_{ij} values are those with the greatest asymmetry in transition probabilities, indicating a directional preference in user transitions between genres.
A small section of the output of this analysis is shown as follows:

```
Animation|Children|Comedy -> Comedy
Drama|War -> Comedy
Drama|Fantasy|Thriller -> Action|Adventure|Sci-Fi|Thriller|IMAX
Action|Adventure|Sci-Fi|Thriller|IMAX -> Drama|War
Action|Adventure|Sci-Fi|Thriller|IMAX -> Action|Sci-Fi|Thriller
```

These examples suggest that users are more likely to move from the left genre group to "Comedy" than from "Comedy" back to those genre groups.

3.0.8 Conclusion

By studying the theory of Markov Graphical models and exploring the research on the prediction of users preferences, we constructed and implemented an adaptation of the Preference Transition Model, applying it to different dataset and with some modification. This allowed us to gain insight into the pathways users commonly follows as they navigate between genres over time, highlighting the functionality of recommendation algorithm, the systems which are behind the majority of the applications people use today.
Overall, these findings provide a comprehensive view of genre dynamics, informing both the design of recommendation strategies and the understanding of people behavior. The PTM framework resulted to be an effective method in capturing the complexity of user transitions, and this approach can be extended to other domains where sequential preferences are of interest.

Bibliography

[1] Dimitri Bertsekas and John N Tsitsiklis. *Introduction to probability*, volume 1. Athena Scientific, 2008.

[2] NetworkX developers. Drawing a directed graph. `https://networkx.org/documentation/stable/auto_examples/drawing/plot_directed.html`, 2023. Accessed: 2023-10-27.

[3] GroupLens. Movielens 25m dataset. `https://grouplens.org/datasets/movielens/25m/`, 2023. Accessed: 2023-10-27.

[4] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[5] Trevor Hastie, Robert Tibshirani, Jerome Friedman, et al. The elements of statistical learning, 2009.

[6] Dimitrios Katselis, Zeyu Zhou, and Lucas Buccafusca. Lecture 1: Markov chains-part i. Technical report, University of Illinois at Urbana-Champaign, 2019. ECE586 MDPs and Reinforcement Learning.

[7] PyTorch. torch.optim.adam. `https://docs.pytorch.org/docs/stable/generated/torch.optim.Adam.html`, 2023. Accessed: 2023-10-27.

[8] Francesco Sanna Passino, Lucas Maystre, Dmitrii Moor, Ashton Anderson, and Mounia Lalmas. Where to next? a dynamic model of user preferences. In *Proceedings of the Web Conference 2021*, WWW '21, page 32103220, New York, NY, USA, 2021. Association for Computing Machinery.

[9] SciPy. scipy.optimize.minimize_scalar. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize_scalar.html`, 2023. Accessed: 2023-10-27.

[10] Larry Wasserman. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013.