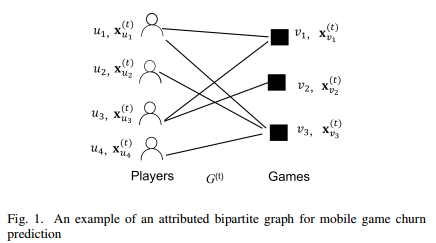
# Semi-supervised and inductive embedding model for churn prediction of large-scale mobile games

1. Introduction

* Previous studies on mobile game churn prediction using traditional ML models (e.g., logistic regression, random forests, Cox regression)
* Limitations:
* Developed for predicting churn of one or a few mobile games -> none is capable of handling the churn prediction of large-scale mobile apps and users
* User-app interaction data often comes with rich contextual info (e.g., wifi connection status, screen brightness, audio volume) >< not considered in previous studies
* Rely on handcrafted features -> usually cannot scale well in practice
* Contributions:
* First solution for **churn prediction of large-scale mobile games** using hundreds of millions of user-app interaction records
* Novel **semi-supervised and inductive model based on embedding** – capture the dynamics between users and mobile games based on introduced temporal loss in the formulated objective function. Model is able to **embed new users or games not used in training**
* Attributed **random walk technique** -> **sample contexts of edges** in an attributed bipartite graph that takes into account both topological adjacency and attribute similarities
* Comprehensive **experimental evaluation with large-scale real-world data** collected from Samsung Game Launcher

1. **Problem formulation**

* Churn: player stopping using a game within a given period (i.e., no app usage in the period). Duration T of the period may vary (e.g., T= 14 or T=30)
* In this paper, we consider the **generic game churn prediction problem without assuming any particular value of T**
* Note: uninstall is different from churn (there may be a large time gap between cessation of playing and uninstall, if any)
* Relationship between players and games – represented by an attributed bipartite graph



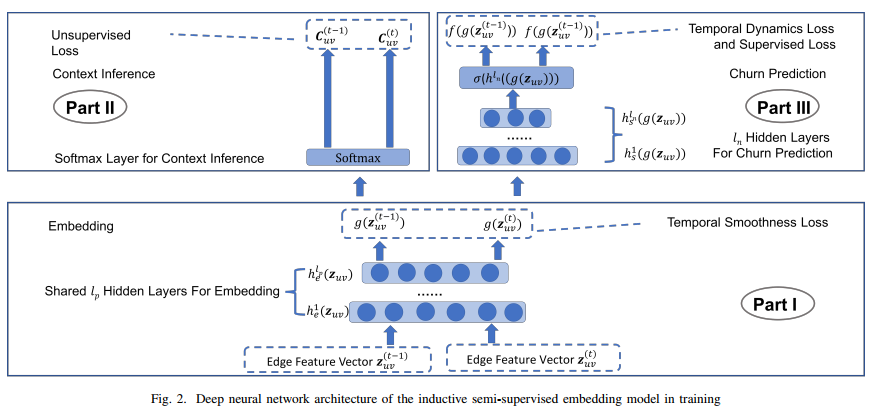
* **Notations**

|  |  |
| --- | --- |
| **Notations** | **Descriptions or Definitions** |
|  | Attributed graph at time |
|  | Attributed graph at time |
|  | Historical attributed graphs |
|  | Set of all player nodes in |
|  | Set of all game nodes in |
|  | Set of all edges in |
|  | Indicator of the existence of edge () in |
|  | Feature vector of user |
|  | Feature vector of game |
|  | Aggregated feature vector of edge |
|  | Number of attributes in |
|  | Number of attributes in |
|  | Embedding dimension |
|  | The edge embedding function |
|  | The churn prediction function |
|  | Number of prediction hidden layers |
|  | Number of embedding hidden layers |

* Let be the attributed bipartite graph at time
* Vertex set denote the set of users
* Vertex set denote the set of games
* Player represented by a node
* Game represented by a node
* Each user is associated with a feature vector , where is the size of
* Each game is associated with a feature vector , where is the size of
* There is an edge between nodes if player has played in the time window
* Set of edges denoted by
* Definition 1: Mobile game churn prediction
* Consider **collection of attributed bipartite graphs observed from time** , which is denoted by
* Let be the **indicator of the existence of edge** in
* **For any edge** , predict the prob. , that is the **probability that**

1. **Methods**
2. **Overview of solution**

* Recent progress in DL and graph embedding, a natural promising direction is to adopt graph embedding frameworks for churn prediction
* **Key technical challenges:**
* All existing methods are transductive -> cannot produce embeddings for new player-game pairs
* Existing methods are either purely supervised or unsupervised -> do not take full advantage of relevance between embedding and a task
* Existing methods are node-centric -> not directly applicable to edge related tasks
* Existing methods only handle a static graph and do not incorporate graph dynamics in embedding >< new players, new games, new player-game relationships every day
* **Propose: a novel inductive semi-supervised embedding model** in dynamic graphs that jointly learns the **prediction function f** and the **embedding function g**



* **DNN consists of 3 parts:**
* **Part I: produce embedding feature vectors** from raw edge feature vectors , where is the size of raw edge feature vectors
  + To learn the prob. of churn, we need to construct a feature vector for each
  + It is impractical to calculate features for all possible edges which may appear in the prediction period because the number of possible edges is large, which is
  + Instead, we **construct the feature vector** **from attribute-wise cosine similarity aggregation of similarity aggregation** of
* **Part II: infer contexts from embedding feature vectors**
  + Context of an edge – the edges that are **similar to and co-occur with the edge under some graph sampling strategy** (e.g., random walk)
  + Part I and II – unsupervised component of the model. Jointly trained by minimizing the **error due to incorrect context inference and inconsistency with temporal smoothness** (III-C for explanation)
  + Part I and II – trained in an inductive and edge-centric way – learn an **embedding function that generalizes to any unseen edges** as long as their feature vector is available
    - Novel attributed random walk to sample similar edges as contexts (III-D)
* **Part III: supervised churn prediction task from embedding feature vectors**
  + Trained by minimizing the error of incorrect churn predictions
* Supervised and unsupervised component are simultaneously trained in **a single objective function**
* Part III and Part II share the **common hidden layers** in part I -> latently coupled with each other -> helps embedding align with the supervised prediction task
* Part I and III both consider **graph dynamics** in training
* Part I: handle graph dynamics by requiring the **embeddings of the same edge at 2 consecutive timestamps to stay close**
* Part III: handles graph dynamics by requiring the **churn prob of the same edge at 2 consecutive timestamps to follow a decaying pattern**
* **Objective function composed of 4 parts:**

(1)

Where:

1. Static Loss Functions

* **k-th hidden layer for churn prediction** (prediction hidden layer)
* Model churn prediction by  **hidden layers**
* **Prediction output layer** represented by (2):

Where:

* **Supervised loss** (3):
* Part II: **embed handcrafted features** **into a latent space**  where m is the size of latent space
* **K-th hidden layer for embedding** (embedding hidden layer)
* **layers in part I to represent embedding function g**
* **Embedding output layer**: (4)
* **Unsupervised loss function**:

Where:

* + . Contextual edges are obtained by attributed random walk on the bipartite graph (discussed later)
* Likelihood of having a contextual edge conditional on the embedding of

(6)

* Weights for regularization consists of
* Regularization part can be expressed as:

(7)

Where are trade-off weights on different regularization terms