February 2022

Graph-based Feature-Serving for Privacy-Aware Machine Learning

Anonymous

Graph-based Feature-Serving for Privacy-Aware Machine Learning

ABSTRACT

This disclosure describes graph-based privacy-aware platforms for serving features to machine learning models, optimized to meet constraints of privacy, model quality, multi-granularity, incremental update, overlapping features, scale, etc. Privacy is preserved by grouping features into a collection, subject to lower-bound constraints, such that features corresponding to individual users are rendered unrecoverable. Such feature aggregation does not prevent generation of personalized recommendations, as model quality is maintained by aggregating similar, rather than arbitrary features, subject to upper-bound constraints. Upper and lower bounds can vary to account for differing privacy sensitivities and feature importance with respect to a given model (multi-granularity). Features can belong to multiple clusters (overlapping features). Incremental feature grouping is used to enable efficient dynamic feature grouping. Feature serving, as described herein, is of low complexity, such that features can be served to a large number of clients at minimum latency without degraded performance.

KEYWORDS

- Machine learning
- Feature serving
- Feature engineering
- Feature query
- Privacy constraint
- Multi-granularity
- Graph clustering
- Cluster hierarchy
- k-anonymity
- Personalization

BACKGROUND

Machine learning (ML) plays an important role in several products, e.g., recommendation systems (which serve relevant content to users); personalization systems (which capture user behaviors and preferences); integrity systems (which identify misinformation and dangerous content in social media); etc. The life cycle for building ML models typically comprises data collection/preprocessing, feature engineering, model training and evaluation, and modeling serving/inference. Feature engineering, which is the identification of inputs to ML models to enable machine-based classification, embedding, prediction, detection, etc. is critical to performance.

Feature-serving platforms provide features to ML models either for training or, during operation, for user queries. For a feature-serving platform to operate at scale, the feature service has to be effective, efficient, and customizable for various models at large scales. Privacy awareness further requires the fulfillment of various constraints when serving features, such that individual user’s information is not leaked even as model quality is preserved. It remains a challenge to build an effective, privacy-aware, ML feature-engineering platform that can operate at scale. Many useful features of ML models for recommendation systems are generated from user-item interactions, e.g., users clicking advertisements, purchasing products, sharing posts, commenting on videos, etc. Such interactions are naturally represented as graphs.

DESCRIPTION

This disclosure describes graph-based, privacy-aware platforms for serving features to ML models, optimized to meet certain requirements and constraints as follows.

Privacy

To prevent leakage of individual user’s information, privacy sensitive features/entities are used collectively. *k*-anonymous features, where at least *k* features or entities are grouped into a collection, provide privacy protection. A higher value of *k* (e.g., 1000) makes it hard to model the individual behavior of a user in the collection. The minimal size constraint, e.g., 200, 1000, etc., is referred to as the lower bound constraint of a feature/entity.

Model quality

While the collective usage of features fulfills requirements of privacy awareness, the aggregation of information smooths out each individual's personality, making personalized recommendations difficult. To mitigate the impact of feature aggregation, similar features are grouped together rather than arbitrary ones. However, for model quality reasons, the number of features grouped together is limited to an upper bound constraint.

Multi-granularity

Due to differing privacy sensitivities and feature importance with respect to a given model, the upper and lower bounds can vary from case to case. This implies multi-granularity for grouping features.

Incremental update

Features can be dynamic. For example, similarity between two videos can manifest in terms of the number of users visiting both. However, such co-engagement information varies with time. To ensure that similar features are grouped together, feature groupings are dynamically updated, which can incur considerable efficiency losses, especially in large scale installations or with features with large numbers of dimensions. Incremental feature grouping, as described herein, can enable efficient dynamic feature grouping. It is worth noting that the incremental update also takes into account stability of the grouping; there are disruptive changes of the group, as it can hurt model quality.

Overlapping features

Features can belong to multiple clusters. For example, if users are clustered into football lovers and swimming lovers, there is a group that loves both sports. While grouping similar features together, provisions are made for a feature to belong to multiple groups.

Scale

Feature serving, as described herein, is of low complexity, such that features can be served to a very large number of clients at minimum latency and without degraded performance.

Graph-based privacy-preserving feature server

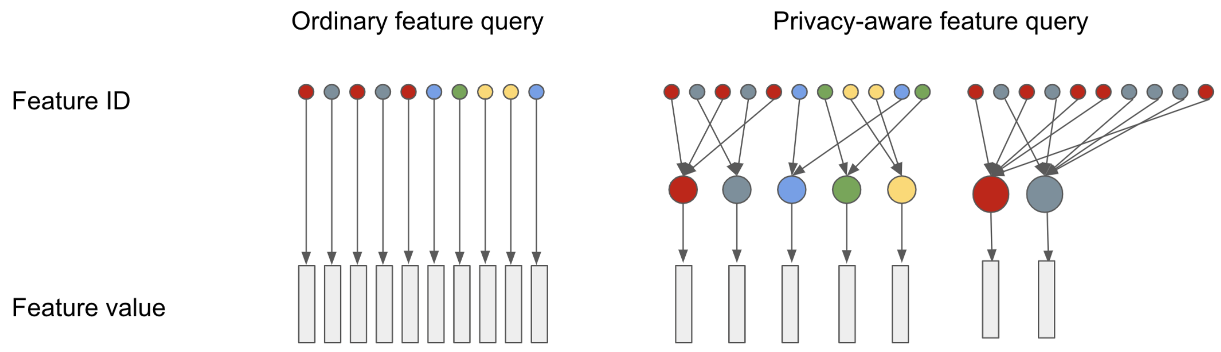

Fig. 1: Graph-based feature serving for privacy-aware machine learning

A graph-based privacy-preserving feature server is illustrated in Fig. 1. Inputs to graph-based feature-serving include:

- A set of feature IDs (circles in Fig. 1), where each identifier is an integer.
- A set of feature values (bars in Fig. 1), where each value is associated with a feature ID. The value can be of any form, such as a numeric weight or an embedding vector.
- A privacy constraint that leads to a set of features being clustered together and sharing the same value, such that the exact value of individual features is not restorable.

    - The strength of the privacy constraint can vary, resulting in different granularity of the aggregated features.

- A set of ML models/algorithms registered in the system, which consumes a subset of features processed in the system
    - A model is a computation on a series of features: $f(fid_1, fid_2, \ldots|\Theta)$, where fid_i is a feature ID, and Θ is the model parameter that will be estimated during training and used during inference.

Some elements of graph-based feature-serving are used for constructing the feature graph and performing hierarchical clustering; some are used for incrementally updating the features; and some are used for feature serving. The output is a privacy-aware feature service to ML models. Specifically, given a registered ML model, when the model requests a feature value by providing the feature ID, the graph-based feature server provides a privacy compliant feature value corresponding to the request.

Architecture

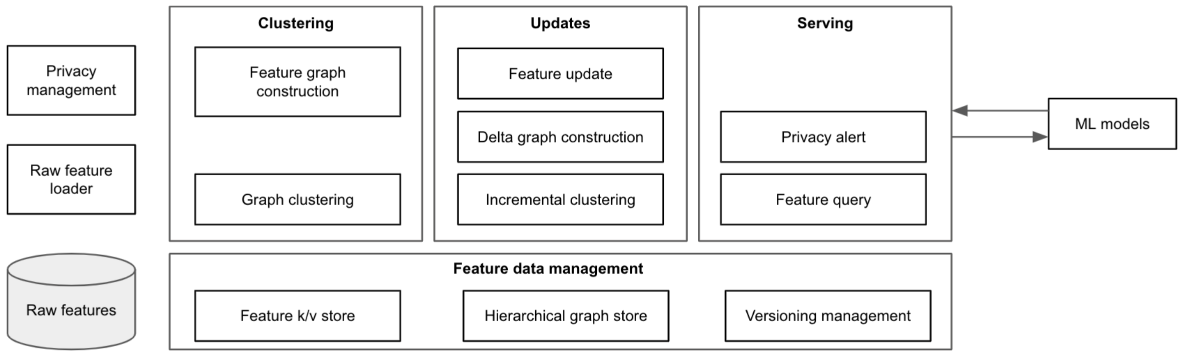

Fig. 2: An example architecture

Fig. 2 illustrates an example architecture for graph-based privacy-preserving feature serving. The architecture can comprise the following modules and sub-modules.

- *Feature data management*: This is a data layer that stores and manages the feature mapping with privacy awareness.
    - Feature key-value store: This component is a key-value store that maps a feature ID to a cluster. Specifically, it maps to the cluster stored in the bottom layer of the hierarchy. When a composite key is provided, such as for addressing the map <*feature_id*, *level*> → *feature_value*, the system first fetches the cluster stored in the bottom layer, and then moves up to a higher layer according to the parameter *level*.
    - Hierarchical graph store: This component stores the result of the graph hierarchical clustering. The bottom layer gives the fine-grained clusters. The clusters merge as the layers go up. Each cluster stores the (aggregated) feature value.
    - Version management: This component maintains different versions of clustering for concurrent access and possible transaction rollback.
- *Clustering*
    - Feature graph construction: this component builds a graph from the raw features. For a given type of feature, say *User*, and a specified affinity measurement (e.g., the number of co-watched videos, or the cosine similarity of the *User* profile embeddings), this component links similar feature IDs to form a graph.
    - Graph clustering: This component performs hierarchical clustering on the constructed graph. The bottom layer gives the finest granularity clusters (e.g., usually a set of smaller clusters) and the smaller clusters further merge to form larger clusters at higher levels. Constraints can be added to ensure that the minimum cluster sizes are greater than a threshold.

- *Updates*
  - Feature updates: Feature updates include adding new features, removing expired features, or updating the affinity between two features.
  - Delta graph: This is a batch mode for feature updates. A representative delta graph is constructed for all the added/modified nodes and edges.
  - Incremental clustering: absorbing a delta graph and updating clustering results.
- *Serving*
  - Feature query: This component returns a feature value given a feature ID and the level of the hierarchy to access. This can be represented as: (*feature_id*, *level*) → *feature_value*.
  - Privacy alert: If a valid feature value cannot be found for a given feature ID, e.g., due to improper privacy requirements, this component issues warnings.

<u>*Workflow*</u>

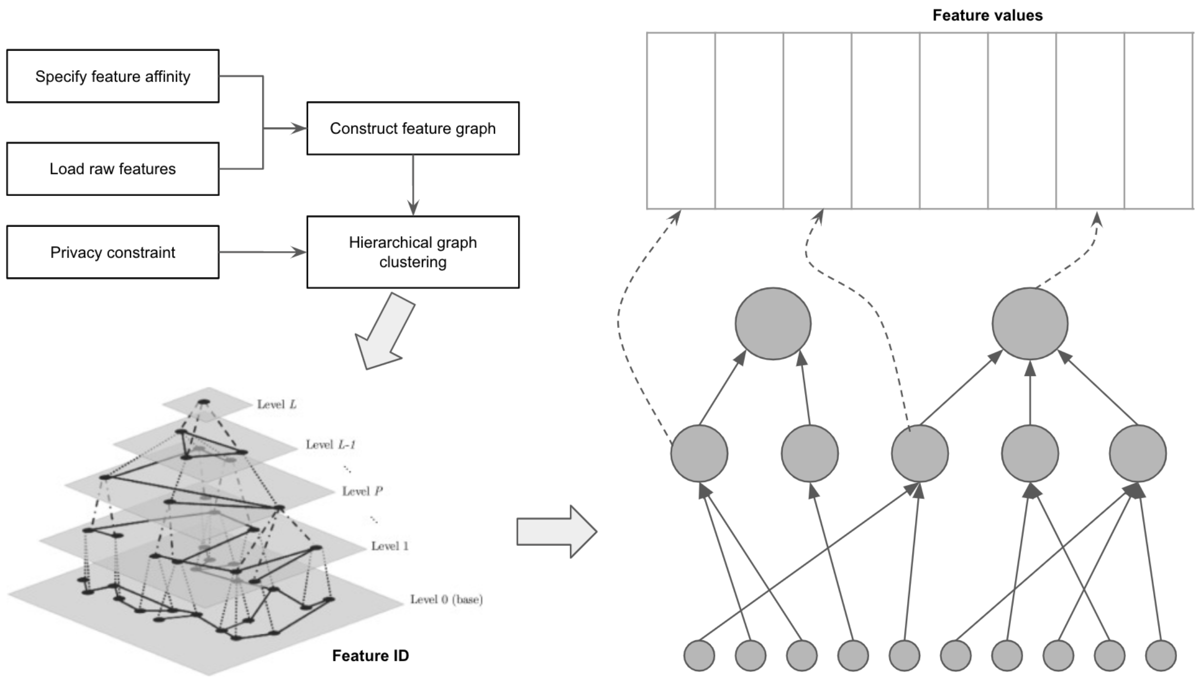

Fig. 3: An example workflow

Fig. 3 illustrates an example workflow of the working of the clustering module.

- The feature graph is constructed based on the loaded raw features and the specified feature affinities. The feature graph is a graph of feature IDs, where each node represents a feature ID and the similar feature IDs are connected by edges. The similarity of feature IDs is specified by the feature affinity, depending on the semantics of the feature ID. For example, if the feature IDs are the *Users*, then the affinity can be defined through engagement (e.g., the number of co-watched videos) or the content (e.g., the cosine similarity of the user profile embedding vectors).
- A hierarchical clustering is performed on the constructed graph, for example, using algorithms such as Louvain and METIS. The privacy constraint is utilized to guide the clustering process. For example, the minimum size of a cluster can be 200 features or more, such that individual feature values are unrecoverable.

Clustering produces a hierarchy of clusters. The bottom layer gives a set of relatively small clusters, which are merged into bigger ones proceeding upwards to the higher levels. A cluster points to a feature value representing the aggregated characteristics of the features within that cluster. For a particular model, when a feature ID is provided, the graph-based privacy-preserving feature-server maps the feature ID to the finest clusters; follows the hierarchical relationship to move up until it reaches a level where the cluster sizes satisfy the privacy requirements for that model; and returns the feature value corresponding to that cluster.

In this manner, the techniques of this disclosure preserve privacy during feature engineering and feature serving for machine learning. Per the techniques, features are not used individually but only in the aggregate, in compliance with constraints. Privacy protection is achieved by providing *k*-anonymity in user features, wherein at least *k* users are grouped into the same bucket, with a higher *k*-value implying a lower risk of identifying a user and their information from the data.

CONCLUSION

This disclosure describes graph-based privacy-aware platforms for serving features to machine learning models, optimized to meet constraints of privacy, model quality, multi-granularity, incremental update, overlapping features, scale, etc. Privacy is preserved by grouping features into a collection, subject to lower-bound constraints, such that features corresponding to individual users are rendered unrecoverable. Such feature aggregation does not prevent generation of personalized recommendations, as model quality is maintained by aggregating similar, rather than arbitrary features, subject to upper-bound constraints. Upper and lower bounds can vary to account for differing privacy sensitivities and feature importance with respect to a given model (multi-granularity). Features can belong to multiple clusters (overlapping features). Incremental feature grouping is used to enable efficient dynamic feature grouping. Feature serving, as described herein, is of low complexity, such that features can be served to a large number of clients at minimum latency without degraded performance.