Recommendation System

Candidate Generation

Comparison

- Memory-Based Collaborative Filtering: [适用] dense interaction data
- Matrix Factorization: [适用] sparse interaction data
- Two-Tower Model: [适用] rich user and item features
- Item2Vec: [适用] focus on item similarity, user behavior sequences is important, such as clickstreams or purchase behaviors
- DeepWalk: [适用] users or items naturally form a network structure, such as a social graph or a knowledge graph

1. Memory-based Collaborative Filtering

核心思想

If two users exhibit similar behaviors, they are likely to enjoy the same items (User-based CF).

If two items are liked by similar users, these items are likely to be similar (Item-based CF).

架构

Input: a user-item interaction matrix, user u and item i whether has interaction (e.g., rating or click)

训练

1. Build the "User-Item" Interaction List: for each user ID, retrieve recently interacted items
2. Build the "Item-Item" Similarity List: (1) Use cosine similarity to calculate the item similarity

   (2) For each item, find its Top-K most similar items

推断

1. Retrieve user’s historical behavior using "User-Item" interaction list
2. Find similar items using "Item-Item Similarity"
3. Interest Score Calculation $\sum_j like(user,\ item_j) \times sim(item_j, item)$.
4. Ranking and returning

优缺点

1. [优] Strong real-time capability
2. [缺] Scalability issues: High computational cost due to pairwise similarity calculations.

   [解决] approximate nearest neighbors (ANN) techniques (e.g., k-d trees, locality-sensitive hashing)
3. [缺] Suffer from data sparsity

2. Matrix Factorization

核心思想

Decomposes the user-item interaction matrix into two low-dimensional matrices (user vectors and item vectors), identifying representations of users and items in a latent space.

架构

1. Input: sparse user-item interaction , approximated by the product of two low-rank matrices

2. Model Training: minimizing the error between the predicted $\hat{R}_{u,i}$ and the observed scores $R_{u,i}$
   Loss: regularized MSE $\mathcal{L} = \sum_{(u,i)\in\text{Observed}} (R_{u,i} - P_u \cdot Q_i^T)^2 + \lambda(\|P_u\|^2 + \|Q_i\|^2)$

优缺点

1. [缺] Struggles to learn complex feature interactions. [解决] add deep learning (e.g., NeuMF)

3. Factorization Machine (FM)

核心思想

FM learns low-dimensional embeddings for features and models pairwise interactions, capturing latent relationships.

架构

1. Input: a sparse vector that includes user, item, and contextual information. (categorical: one-hot)
2. Formular：$\hat{y} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$

> - w_0 is the global bias, w_i is the weight for the i_{th} feature, capturing its individual contribution.
> - $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} v_{j,f},$ dot product of the latent vectors for features i and j, capturing their second-order interactions.
> - $\hat{y}$ is the predicted value, representing either a user's preference for an item or the probability of a certain interaction.

Loss: Regugarlization MSE: $\mathcal{L} = \sum_{(x,y)} (y - \hat{y})^2 + \lambda(\|w\|^2 + \|\mathbf{v}\|^2)$

优缺点

- [优] Robustness to sparse data
- [缺] Limited to second-order feature interactions. [解决] Add deep learning (e.g., DeepFM)

4. Two-Tower Model

核心思想

Learn embeddings for users and items separately using two independent models, and make recommendations based on embedding similarity.

架构

1. Input: User features and item features:
2. User Tower (Item Tower): (1) Takes user (item) features, embed categorical features into dense vectors. (2) Embedding vectors + numerical features fed into a DNN, generate the representation
3. Similarity Calculation

训练

- Maximize/minimize the similarity score for user-item positive/negative pairs.
- Loss: Binary Cross-Entropy Loss or Contrastive Loss

> $\mathcal{L} = -\sum_{(u,i,j)} \log \sigma(\mathbf{u} \cdot \mathbf{v}_i - \mathbf{u} \cdot \mathbf{v}_j)$ where i is a positive item and j is a negative item.

优缺点

- [缺] Loss of interaction information

5. Item2Vec

核心思想

Use sequences of user interactions (e.g., purchases, clicks) and their frequencies to capture item-to-item similarities and embeding items into a low-dimensional representations.

架构

1. Input: user-item interaction sequences, such as a sequence of items clicked or purchased by users
2. Using a sliding window, generate pairs from the sequence.

   > For example, with a sequence [A,B,C,D] and a window size of 2, generate pairs such as (A,B),(A,C),(B,C),(B,D),(C,D).

3. Using Skip-Gram model

训练

- Embedding Learning: maximize the embedding similarity between positive pairs

  > Positive pair are generative using the sequence: the target item and its context items.

- Loss: $\mathcal{L} = -\log \sigma(\mathbf{v}_{\text{target}} \cdot \mathbf{v}_{\text{context}}) - \sum_{i=1}^{N} \log \sigma(-\mathbf{v}_{\text{target}} \cdot \mathbf{v}_{\text{negative},i})$
- After training, each item has a fixed embedding vector, which used to calculate similarity and recommendation.

优缺点

- [缺] Lack of user personalization: only models relationships between items and ignores user-specific preferences. [解决] Incorporate user features (e.g., User2Vec)

6. DeepWalk

核心思想

DeepWalk generates sequences through random walks on a graph, and then it leverages the sequences to capture node-to-node similarities, and embedding nodes into a low-dimensional representations.

架构

1. Input: a social network or graph data, where nodes represent users (or item) and edges represent relationships (e.g., friendships or co-occurrences):
2. Perform random walks on the graph to generate node sequences. A random walk starts from a node and randomly selects a neighboring node at each step, producing sequences similar to "sentences." For example, a random walk might produce [A,B,C,D,E].
3. Skip-Gram model

训练

- Embedding Learning: maximize the embedding similarity between neighboring nodes = maximize the conditional probability of context nodes given a target node: $\prod_{u \in V} \prod_{v \in \text{Context}(u)} P(v|u)$
- Loss: $\mathcal{L} = -\log \sigma(\mathbf{v}_u \cdot \mathbf{v}_v) - \sum_{i=1}^{N} \log \sigma(-\mathbf{v}_u \cdot \mathbf{v}_{\text{negative},i})$
- After training, each node is represented by an embedding vector.

优缺点

- [缺] Limited global information: Only captures local graph structures, missing global structural patterns. [解决] add GNNs (e.g., GCN or GraphSAGE)
- [缺] Simple random walk strategy: Existing random walk strategies may not fully capture diverse relationships between nodes. [解决] use biased random walks (e.g., Node2Vec).

Speed Up: Approximate Nearest Neighbor

To speed up the candidate generation process, ANN can be used.

- ANN groups embeddings of similar items into the same bucket. During a query, it compares the query embedding with the representative vector of the bucket or directly searches the bucket to quickly retrieve similar items.
- Locality-Sensitive Hashing uses specific hash functions to map embeddings of similar items into the same hash bucket. During a query, it computes the hash value of the query embedding to directly locate the corresponding bucket and quickly retrieve similar items from it.
- Summary
  1. 数据维度：
  - 低维：树结构，例如 K-Dimensional Tree
  - 高维：(1) 图结构，例如 Hierarchical Navigable Small World
    (2) 量化方法，例如 Product Quantization
  2. 数据规模：
  - 小规模：(1) 哈希， 例如 Locality-Sensitive Hashing (2) 树结构，例如 K-Dimensional Tree
  - 大规模：(1) 图结构，例如 Hierarchical Navigable Small World
    (2) 量化方法，例如 Product Quantization
  3. 精度要求：
  - 高精度：(1) 图结构，例如 Hierarchical Navigable Small World
    (2) 深度学习，例如深度嵌入与近邻搜索（Deep Embedding + Nearest Neighbor Search）
  - 快速近似：(1) 哈希， 例如 Locality-Sensitive Hashing (2) 投影，例如 Random Projection

Similarity Measure

- Cosine Similarity measures directional similarity of vectors, suitable for high-dimensional sparse data
- Pearson Correlation measures linear correlation between vectors, suitable for continuous numerical data with linear relationships
- Euclidean Distance measures absolute straight-line distance between two points, suitable for low-dimensional continuous data, especially when features are normalized

Sort

Comparison

- Factorization Machine (FM): [适用] optimizing a specific metrics (e.g., CTR, conversion rate) + feature interactions play a crucial role .
- Wide & Deep Model: [适用] both memorization (capturing frequent past patterns) and generalization (identifying new relevant recommendations) are important
- Deep Interest Network (DIN): [适用] user behavior sequences is important
- Graph Neural Network (GNN): [适用] users or items naturally form a network structure

1. Wide & Deep Model

核心思想:

The Wide & Deep Model consists of two components: the Wide part and the Deep part. The Wide part captures explicit feature interactions to retain past user preferences, while the Deep part uncovers high-order feature interactions, identifying latent needs that users may not have explicitly expressed. By combining these two, the model enhances both accuracy and novelty in recommendations.

For Example, in an e-commerce scenario, the wide part leverages a user's purchase history to reinforce recommendations based on known preferences. Meanwhile, the Deep part analyzes implicit signals to identify and suggest new products the user might be interested in.

架构

1. Wide part: Input: categorical features
    - Cross-product layer: generate feature crosses that capture explicit feature interactions.
    - Linear layer: process interactions, generate wide output
2. Deep part: Input: both numerical and categorical features
    - Embedding layers: converting categorical data into dense embeddings
    - DNN: process concatenated embeddings and numerical features, generate deep output
3. Weighted linear layer: Input: combined wide and deep outputs.
    - The outputs from both parts are then combined and passed through a weighted linear layer, followed by an activation function (e.g., sigmoid) to produce the final prediction.

训练

- Pointwise approach: data point: user-item pair and a label.
- Minimize the difference between the predicted and actual labels.
- Loss: cross-entropy loss / MSE if label is a binary class/number.

优缺点

- [缺] Wide component depends on feature engineering, requiring extensive manual feature design, increases development costs. [解决] Use automated feature crossing techniques (e.g., CrossNet in Deep & Cross Networks) .

2. Deep Interest Network (DIN)

核心思想

DIN uses an attention mechanism to emphasizes recent actions to better capture the user's real-time preferences.

Example: If a user recently browsed several smartphones, DIN will prioritize those browsing behaviors most similar to the current candidate product, enhancing recommendation precision.

架构

1. Input: (1) User historical behavior sequence (e.g., items the user clicked, viewed, or purchased) (2) target item features.

    > The goal is to predict the user's interest (e.g., click probability or purchase probability) in the target item based on their historical behavior.

2. Embedding layer: Each item in the user's historical behavior sequence is embedded into a dense vector representation
3. Attention layer: According to its relevance to the target item, assigns importance weights to each historical item, Generating weighted behavior features.

    > For example, if the target item is an "electronic device," the model will focus more on historical behaviors related to "electronics."

4. Deep network: The weighted behavior features are concatenated with the target item features. Concatenated fed into a DNN. Output an interest score

- Loss: cross-entropy loss (classification task)

优缺点

- [缺] Limited representation of diverse interests: especially when their preferences shift frequently.

  [解决] Add multi-interest modeling (e.g., DIEN)

3. Graph Neural Network (GNN)

核心思想

For each node, GNN aggregates features and relationships from its “neighbors” (other users or items directly connected to it) to learn an embedding representation. Recommendation systems utilize these embeddings to predict the similarity between nodes.

> GNNs perform “message passing” to update node features and learn richer representations of the graph.

架构

Suppose we have a user-item interaction graph, where nodes represent users or items, and edges represent interactions such as clicks, ratings, or purchases:

1. Input
    - A user-item interaction graph, where nodes represent users or items, and edges represent interactions such as clicks, ratings, or purchases
    - Node features: Attributes of users and items (e.g., user age, gender; item category, price).
2. Representation Learning: GNNs perform “message passing” to update node features and learn richer representations of the graph.
    - Message Passing: Each node collects information from its neighboring nodes.
    - Feature Update: The collected information is combined with the node’s own features to generate a new representation. This process is repeated over multiple layers.
    - Simply put, it’s like each node “consulting” its neighbors and using their insights to improve its own representation.

训练

- The goal is to learn high-quality embeddings for users and items, which can be used to predict interaction probabilities.
- Loss depends. (1) binary classification (click prediction): Cross-entropy (2) rating prediction: MSE
- After training, each node has a high-dimensional embedding vector.

优缺点

- [缺] Over-smoothing problem: As the number of GNN layers increases, node features tend to become overly similar, leading to a loss of distinctiveness. [解决] Use residual connections (e.g., Res-GNN) to preserve unique node information.

Multiple Goals

Training loss is a weighted sum of the losses from each task.

1. Multi-Task Model

核心思想

Muti-task model uses a shared model to capture the fundamental relationship between users and items, generating a shared feature representation. This shared representation is then used by seperate modules to predict metrics for each task .

架构

The model has two main parts:

- Shared Layer: (1) Input: concatenation of all features (2) Outputs: a representation vector
- Task-Specific Layer: (0) one task, one NN. (1) Input: representation (2) Outputs: prediction

2. Multi-Gate Mixture-of-Experts (MMoE)

核心思想

MMoE uses several independent "expert modules" to capture different patterns or characteristics of the data, generating multiple feature representations. In MMoE, each task employs a specific mechanism to integrate these representations and makes predictions based on the integrated features.

架构

1. Expert Layer: multiple expert modules = multiple NN, each corresponding to a specific characteristics
   - Each NN Input: concatenation of user features, item features, and context features.
   - Each NN Output: a feature vector that represents the expert's specific understanding of users-items relationship.
2. Gating Network Layer: multiple expert modules = multiple NN, each corresponding to a specific task
   - Each NN Input: concatenation of user features, item features, and context features.
   - Each NN Output: a vector where each value represents the weight assigned to a corresponding expert, used to combine the feature vectors from the expert layer.
3. Task-Specific Layer: multiple prediction modules = multiple NN, each corresponding to a specific task
   - Each NN Input: the task-specific feature vector obtained by weighting the expert layer's outputs.
   - Output for each NN: the predicted value for each task (e.g., click-through rate, conversion rate).

Diversity

MMR vs DPP

Compared to MMR, DPP is more computationally complex but can optimize global diversity.

1. Maximal Marginal Relevance (MMR)

核心思想

MMR is an iterative selection process. At each step, it tries to pick the most relevant item while avoiding redundancy, balancing relevance and diversity.

架构

Suppose we have a candidate list of 100 articles, ranked by relevance. MMR further refines the selection:

1. Initialize: Start with an empty “selected list” and put all articles in the “remaining list.” Pick the article with the highest relevance score and move it to the “selected list.”
2. Iterative selection: Repeat the following steps until k articles are selected:

    - Score calculation: For each article in the “remaining list,” calculate a new score that combines its relevance to the user and its similarity to the articles in the “selected list.” Articles that are too similar to what’s already selected will get lower scores.
    - Select the top-scoring article: Move the article with the highest new score from the “remaining list” to the “selected list.”

2. Determinantal Point Process (DPP)

核心思想

DPP aims to select k items from n candidates such that the selected items have both high relevance and low similarity. DPP uses the mathematical concept of a determinant to quantify the diversity of a set.

架构

Suppose we have a candidate list of 100 articles. Each article has a relevance score and a feature vector that represents its content, such as topic or style.

1. Initialize : Start with an empty “selected list” and put all articles in the “remaining list.” Build a similarity matrix using the feature vectors to measure pairwise similarity between articles.
2. Iterative selection: Repeat the following steps until k articles are selected:
    - Evaluate subsets: For each article in the “remaining list,” calculate how much it improves the quality and diversity of the “selected list” if added. Diversity is measured using the determinant, where a larger value means greater diversity.
    - Select the best article: Choose the article that maximizes the combined quality and diversity of the "selected list," and move it from the "remaining list" to the "selected list."

Data Challenge

Cold Start

- User cold start: (1) basic demographic information (2) onboarding questions
- Item cold start: (1) content-based recommendations (meta data) (2) new items to gather feedback
- System cold start: popularity-based recommendations

Long-Tail Effect

- Problem: few items account for most of the clicks $\rightarrow$ long-tail items are poorly learned.
- Solution: Self-supervised Learning (Contrastive Learning)

  Two different feature vectors (A, B) $\rightarrow$ (feature transformations) $\rightarrow$ two new feature vectors (A', B')

  Training maximizes the similarity between A & A', B & B'，minimizing A & B, A' & B'

> Feature Transformation
>
> - Random Mask： randomly mask some discrete feature
>
>   某物品的特征是$\mathscr{U}$= {数码,摄影}, Mask 后的类目特征是 $\mathscr{U}'$ = {default}。
> - Dropout (only for multi-value discrete feature)：randomly discard 50% values
>
>   某物品的特征是$\mathscr{U}$= {数码,摄影}, Mask 后的类目特征是 $\mathscr{U}'$ = {摄影}。
> - Complementary：suppose an item has 4 features, randomly divide into two groups
>
>   {ID，类目，关键词，城市}: { ID，default，关键词，default }, { default，类目，default，城市 }
> - Mask correlated features
>
>   原始数据：u=女,v=美妆；Mask 后：u=女,v=[MASK]