Content based Filtering

Bài toán

- **Ý tưởng chính vs. lọc cộng tác**
  - *Lọc cộng tác* khuyến nghị dựa trên xếp hạng (ratings) của những người dùng có hành vi tương tự.
  - *Lọc dựa trên nội dung* khuyến nghị dựa trên **tính năng** của người dùng và của món (item). Nó dùng thông tin về *ai* (user features) và *cái gì* (item features) để tìm sự phù hợp.
- **Dữ liệu cần có**
  - Vẫn dùng ma trận xếp hạng $r_{i,j}$ hoặc $y_{i,j}$ (ai đã chấm gì), nhưng bổ sung:
    - Vector tính năng của người dùng: $x_j^{(u)}$ (ví dụ: tuổi, giới tính, hành vi xem, trung bình theo thể loại...).
    - Vector tính năng của item (phim): $x_i^{(m)}$ (ví dụ: năm phát hành, thể loại one-hot, điểm trung bình của phim...).
- **Mục tiêu mô hình**
  - Học hai vector biểu diễn ngầm (latent) cùng kích thước:
    - $v_j^{(u)}$ — biểu diễn sở thích/người dùng.
    - $v_i^{(m)}$ — biểu diễn đặc trưng/món.
  - Dự đoán xếp hạng bằng **tích vô hướng**: $\hat{y}_{i,j} = v_i^{(m)} \cdot v_j^{(u)}$.
  - Vì để nhân chấm, v^{(u)} và v^{(m)} **phải cùng kích thước**, dù $x^{(u)}$và $x^{(m)}$ gốc có kích thước khác nhau.
- **Vấn đề & thách thức**
  - Làm sao dịch từ tính năng quan sát $x^{(u)}$, $x^{(m)}$ sang biểu diễn $v^{(u)}$, $v^{(m)}$ hiệu quả?
  - x người dùng và x item có thể rất khác về kích thước — cần ánh xạ vào cùng không gian latent.

Áp dụng Deep Learning

Ý tưởng tổng quát

- Ta muốn từ **vector đặc trưng của người dùng** (x_u: tuổi, giới tính, quốc gia, ...) tính ra **vector biểu diễn người dùng** v_u.
- Tương tự, từ **đặc trưng phim** (x_m: năm phát hành, diễn viên, thể loại, ...) tính ra **vector biểu diễn phim** v_m.
- Dự đoán xếp hạng của người dùng j với phim i:

D

Kiến trúc mô hình

- **Hai mạng nơ-ron riêng biệt**:
  - *Mạng người dùng*: đầu vào x_u, qua vài lớp dense → đầu ra v_u (ví dụ: 32 chiều).
  - *Mạng phim*: đầu vào x_m, qua vài lớp dense → đầu ra v_m (cùng kích thước).
- Lớp cuối không phải 1 node mà là một vector (ví dụ 32 số).
- Nếu bài toán là nhị phân (like/dislike), ta áp dụng **sigmoid** cho tích chấm để dự đoán xác suất.

Hàm mất mát & huấn luyện

- Xây dựng hàm chi phí:

$$J = \sum_{(i,j)\in\mathcal{D}} \left(v_u^{(j)} \cdot v_m^{(i)} - y_{ij} \right)^2$$

- Tối ưu **toàn bộ tham số** của cả hai mạng bằng gradient descent hoặc các optimizer khác.
- Có thể thêm **regularization** (L2) cho các trọng số để tránh overfitting.

Ứng dụng & lợi ích

- Sau khi huấn luyện, v_m là vector mô tả phim → ta tìm phim tương tự bằng **khoảng cách giữa các** v_m (ví dụ: Euclidean hoặc cosine).

- Danh sách “phim tương tự” có thể **tính trước offline**, giúp hệ thống phản hồi nhanh khi người dùng truy cập.

Nhận xét & thực tế

- Sức mạnh của mạng nơ-ron: dễ kết hợp nhiều mô hình con (ở đây là user network + item network).
- Thành công phụ thuộc rất nhiều vào **thiết kế đặc trưng đầu vào** (feature engineering).
- Với danh mục vật phẩm khổng lồ, việc tính toán có thể đắt đỏ → cần kỹ thuật tối ưu mở rộng (video tiếp theo sẽ đề cập).

Tính mở rộng

Vấn đề

- Với các danh mục rất lớn (hàng triệu–chục triệu phim, bài hát, quảng cáo, sản phẩm...), không thể chạy toàn bộ mạng nơ-ron cho **mọi mục** mỗi khi người dùng xuất hiện → chi phí tính toán quá cao.

Giải pháp hai bước

Bước 1: **Truy xuất (Retrieval)**

- Mục tiêu: nhanh chóng lấy **một danh sách nhỏ** (100–500 mục) từ kho khổng lồ, sao cho trong danh sách có *các ứng viên tiềm năng tốt*.
- Một số kỹ thuật:
  - Tìm các item “tương tự” với những phim gần đây user đã xem (dựa trên khoảng cách giữa các vector).
  - Lấy top-N trong những thể loại user thích nhiều nhất.
  - Lấy top-N ở quốc gia của user.
- Các danh sách này có thể **tính trước offline** (precompute similarity, top items by genre...).
- Chấp nhận rằng danh sách có thể lẫn mục “không hay”, miễn sao bao phủ đủ những lựa chọn tốt.

Bước 2: **Xếp hạng (Ranking)**

- Đưa tất cả ứng viên từ bước 1 vào **mô hình dự đoán**:
  - Lấy v_u (embedding của user hiện tại).
  - Lấy v_m (embedding của từng item trong danh sách).
  - Tính $\hat{y}_{ij} = v_u \cdot v_m$ (hoặc thêm head mạng để dự đoán xác suất/rating).
- Sắp xếp danh sách theo điểm dự đoán → chọn top-K cuối cùng.

Tối ưu hóa hiệu năng

- Nếu đã **tính trước embedding** v_m cho mọi item, khi user truy cập:
  - Chỉ cần chạy mạng user một lần để lấy v_u.
  - Tính tích chấm giữa vuv_uvu và v_m của các ứng viên → rất nhanh.
- Chọn số lượng item trong bước truy xuất (100/500/1000) dựa trên:
  - Độ chính xác mong muốn.
  - Thời gian phản hồi cho phép.
  - Thường thử nghiệm offline để tìm điểm cân bằng.

Lợi ích

- Phân tách **retrieval** & **ranking** cho phép:
  - **Nhanh**: loại bỏ sớm phần lớn item không liên quan.
  - **Chính xác**: mô hình xếp hạng tinh chỉnh trong tập nhỏ.

Vấn đề đạo đức

Bối cảnh chung

- Hệ thống giới thiệu (recommender systems) mang lại lợi nhuận lớn, nhưng đôi khi cũng dẫn đến tác động tiêu cực cho người dùng hoặc xã hội.
- Vấn đề nằm ở **mục tiêu tối ưu** mà mô hình được huấn luyện:
  - Dự đoán phim người dùng đánh giá 5★ → khá lành mạnh.
  - Dự đoán sản phẩm người dùng dễ mua → hợp lý.
  - Nhưng nếu tối ưu "quảng cáo được nhấp nhiều nhất", "lợi nhuận cao nhất", hoặc "thời gian xem tối đa", thì có thể tạo vòng phản hồi xấu.

Một số ví dụ tiêu biểu

Quảng cáo

- **Vòng phản hồi tích cực** có thể giúp:
  - Doanh nghiệp tốt (vd: du lịch chất lượng) → có lợi nhuận → đặt giá thầu cao → quảng cáo được ưu tiên → phục vụ nhiều người hơn → tiếp tục tăng trưởng (vòng đạo đức).
- Nhưng cũng có thể khuếch đại:
  - Doanh nghiệp gây hại (vd: cho vay "payday loan" lãi suất cực cao) → lợi nhuận lớn nhờ bóc lột → đấu giá quảng cáo cao → được ưu tiên hiển thị → thu hút thêm nạn nhân (vòng độc hại).

Tối đa hóa thời gian xem / tương tác

- Mục tiêu "giữ người dùng lâu nhất" khiến thuật toán ưu tiên nội dung gây sốc, thuyết âm mưu, thù hận... vì chúng **hấp dẫn** → người dùng dành nhiều thời gian hơn → nhưng xã hội bị lan truyền thông tin độc hại.

Lợi nhuận vs. lợi ích người dùng

- Một số trang không hiển thị sản phẩm phù hợp nhất mà **sản phẩm có biên lợi nhuận cao**.
- Người dùng thường không biết tiêu chí sắp xếp đó.

Cải thiện & giảm tác hại

- **Loại bỏ / hạn chế** quảng cáo từ doanh nghiệp bóc lột (dù định nghĩa "bóc lột" rất khó, cần thảo luận xã hội).
- **Lọc bớt nội dung xấu**: lời nói thù hận, lừa đảo, bạo lực, thông tin sai lệch... (nhưng việc định nghĩa & thực thi cũng không dễ).
- **Minh bạch** với người dùng: cho họ biết tiêu chí ra quyết định (tối đa lợi nhuận? phù hợp sở thích? an toàn nội dung?).
- Khuyến khích **thảo luận công khai** và lấy nhiều quan điểm trước khi quyết định chính sách cho hệ thống gợi ý.
- Cân nhắc thiết kế mục tiêu tối ưu không chỉ vì lợi nhuận, mà còn **phúc lợi xã hội dài hạn**.

Cài đặt

Ý tưởng tổng thể

- Ta xây dựng **hai mạng nơ-ron riêng biệt**:
  - Mạng **người dùng** → biến các đặc trưng người dùng thành vector nhúng vuv_uvu.
  - Mạng **vật phẩm** (phim/sản phẩm) → biến đặc trưng item thành vector nhúng vmv_mvm.
- Sau đó dự đoán mức độ phù hợp giữa người dùng và vật phẩm bằng **tích chấm** giữa hai vector này.

Kiến trúc mạng

- Sử dụng `tf.keras.Sequential` cho mỗi mạng.
- Ví dụ:
  - Hai lớp `Dense` (ẩn) với kích hoạt `relu`.
  - Lớp cuối cùng xuất ra **32 số** (kích thước embedding).
- Tách riêng pipeline của người dùng và item để dễ bảo trì / tái sử dụng.

Tiền xử lý & chuẩn hóa

- **Trích xuất đặc trưng** của người dùng và item bằng các `Input` layer tương ứng.
- Sau khi mạng tính toán vuv_uvu và vmv_mvm, **chuẩn hóa L2** (`tf.linalg.l2_normalize`) để vector có độ dài 1.

  → Điều này giúp:
  - Ổn định huấn luyện.
  - So sánh hai vector như cosine similarity.

Tích chấm & đầu ra

- Dùng lớp `tf.keras.layers.Dot(axes=1)` để lấy **dot product** giữa vuv_uvu và vmv_mvm.
- Đây chính là dự đoán mức độ “thích hợp” giữa người dùng và item.

Định nghĩa mô hình & huấn luyện

- Khai báo `Model(inputs=[user_features, item_features], outputs=dot_output)`.
- Sử dụng **Mean Squared Error (MSE)** làm hàm mất mát, tối ưu bằng Adam/SGD.
- Huấn luyện trên các cặp (user, item) với nhãn là rating hoặc mức độ tương tác.

Mẹo nhỏ trong mã

- Chuẩn hóa L2 cho cả hai vector là một “trick” đơn giản nhưng giúp mô hình hội tụ và dự đoán ổn định hơn.
- Bạn có thể “chơi” với:
  - Số tầng, số đơn vị ẩn.
  - Loại kích hoạt (ReLU, tanh...).
  - Kích thước embedding (32 chỉ là ví dụ).