Современные подходы в области построения рекомендательных систем

Паша Коваленко, 617 группа

Спецсеминар «Алгебра над алгоритмами
и эвристический поиск закономерностей»

Задача рекомендательных систем

Имеется множество пользователей U и множество объектов I, $|U| = n, |I| = m$. Для некоторых пар (u, i) известна оценка r_{ui}, которую пользователь u поставил объекту i.

Составим матрицу оценок $R = \{r_{ui}\}, u \in U, i \in I$. В матрице есть пропуски.

Требуется заполнить пропуски в матрице, то есть предсказать оценки $\hat{r}_{ui}$, которые пользователи еще не поставили.

$I(u)$ — множество объектов, которые оценил пользователь u
$U(i)$ — множество пользователей, оценивших объект i

Метрики в задаче предсказания рейтинга

Стандартные метрики для задач регрессии — MSE, MAE

В случае бинарного отклика можно использовать метрики качества бинарной классификации — log loss, ROC AUC

AUC логичнее считать отдельно для каждого пользователя и потом усреднять по пользователям

Задача рекомендательных систем

На практике более интересна задача top-N рекомендаций:

Для каждого пользователя нужно предсказать N наиболее релевантных ему объектов.

Попарные метрики

Если объект i_1 релевантнее, чем объект i_2, то потребуем $\hat{r}_{i_1} > \hat{r}_{i_2}$

Пример: $\mathcal{L}(\hat{r}_{i_1}, \hat{r}_{i_2}) = \log(\sigma(\hat{r}_{i_1} - \hat{r}_{i_2}))$ — аналог log loss

Можно использовать в случаях, когда задан только относительный порядок некоторых пар объектов

Метрики ранжирования

Считаются отдельно для каждого пользователя и усредняются по пользователям

$$precision@k = \frac{\#\text{положительных примеров среди первых k}}{k}$$

$$recall@k = \frac{\#\text{положительных примеров среди первых k}}{\#\text{положительных примеров}}$$

Discounted Cumulative Gain: $DCG(\hat{r}) = \sum_{i=1}^{m} \frac{\hat{r}_i}{\log(i+1)}$

— объекты упорядочены по убыванию релевантности

Normalized DCG: $nDCG(\hat{r}) = \frac{DCG(\hat{r})}{DCG^*}, \qquad DCG^* = \max_r DCG(r)$

Проблемы рекомендательных систем

- Неявные оценки
  Оценки — это не всегда и не только явный отклик пользователя, но и сам факт взаимодействия с объектом
- Неслучайность пропусков
  Отсутствие оценки — это скорее отрицательная оценка, чем положительная
- Разреженность данных
  Чаще всего матрица R очень разреженная
- Холодный старт
  Могут появляться как новые пользователи, так и новые объекты, и для них тоже нужно составлять рекомендации
  Здесь же проблема сессионных рекомендаций — рекомендации по первым нескольким действиям на сервисе

Классификация рекомендательных систем

- Content-based filtering
  Даны признаки пользователей и объектов
- Collaborative filtering
  Нет никакой информации о пользователях и объектах, даны только факты взаимодействия между ними
  - Memory-based
    При предсказании в явном виде используются все предыдущие действия пользователей
  - Model-based
    История действий пользователя описывается некоторой моделью

Content-based filtering

Для всех пользователей известен некоторый набор признаков f_u, для всех объектов — набор признаков f_i

Для pointwise метрик можно рассматривать $([f_u, f_i], r_{ui})$ как стандартную задачу регрессии

Существуют модификации стандартных алгоритмов для попарных и ранжирующих метрик

LambdaMART

Используется для оптимизации DCG. Сводит задачу к задаче попарного ранжирования

В основе алгоритма — градиентный бустинг над деревьями. Для каждого пользователя u и каждой пары объектов (i, j) таргет для обучения очередного дерева — это изменение DCG для пользователя u при перестановке i и j местами

Современные подходы

Если перейти к сглаженной оценке метрики качества, то можно использовать весь аппарат нейронных сетей

Пропадает необходимость в генерации признаков. Можно обучать end-to-end архитектуры для работы с изображениями, звуком, текстами и т.д.

Пример: Content-based рекомендации музыки. Для пользователей обучаем скрытое представление, а для песен скрытый вектор получаем из сверток над звуковыми признаками

Memory-based подход (пример)

Можно найти пользователей, похожих на данного, и при рекомендациях для этого пользователя ориентироваться на то, что оценили похожие пользователи. Это можно формализовать следующим образом:

$$\hat{r}_{ui} \sim \sum_{u' \in U \setminus u} r_{u'i} \; sim(u, u')$$

где $sim(u, u')$ — похожесть пользователей u и u', например, корреляция их оценок

Матричные разложения

Один из примеров model-based подхода — матричные разложения

Можно приблизить матрицу $R \in \mathbb{R}^{n \times m}$ в виде $R = PQ^T$, $P \in \mathbb{R}^{n \times k}, Q \in \mathbb{R}^{m \times k}$, k — параметр модели

Та же идея с другой стороны: представим $\hat{r}_{ui} = \langle p_u, q_i \rangle$, где $p_u, q_i \in \mathbb{R}^k$ — скрытые (латентные) представления пользователей и объектов, которые мы обучаем на тренировочной выборке

SVD++

Развитие идеи матричных разложений

$$\hat{r}_{ui} = r + r_u + r_i + \left\langle p_u + \sum_{j \in I(u)} y_j, q_i \right\rangle$$

$r, r_u, r_i \in \mathbb{R}$ — глобальное смещение и смещение оценок пользователя и объекта, $y_i \in \mathbb{R}^k$ — еще одно латентное представление объектов

Factorization machines

Сложная модель, обобщающая идею матричных разложений, но также использующая контентные признаки объектов и пользователей

Представим пару (u, i) в виде вектора вещественных признаков $x = [one\_hot(u), one\_hot(i), f_u, f_i] \in \mathbb{R}^d$

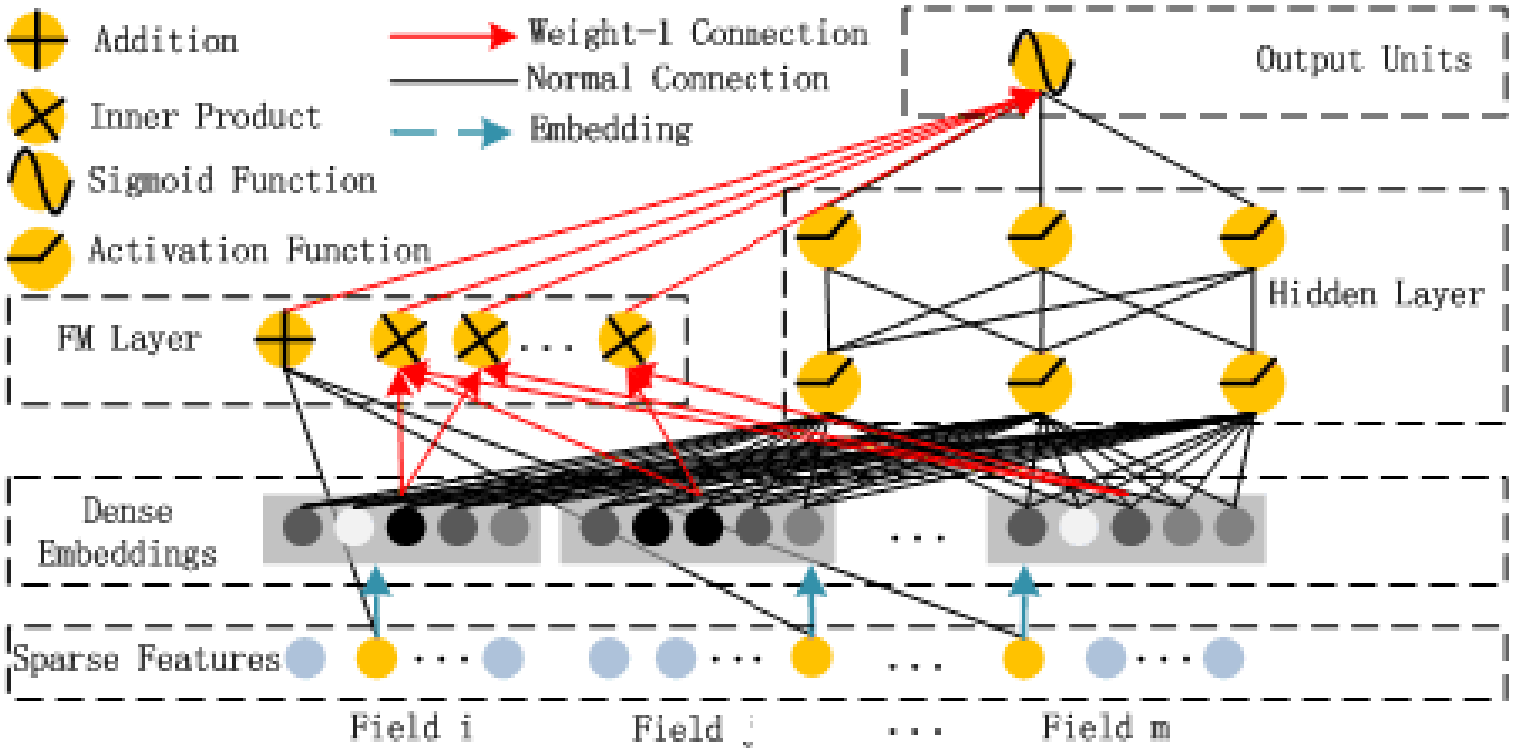
$$\hat{r}_{ui} = r + \sum_{i=1}^{d} x_i w_i + \sum_{1 \le i < j \le d} \langle v_i, v_j \rangle \, x_i x_j$$

Здесь $w_i \in \mathbb{R}$ — веса отдельных признаков в линейной модели, $v_i \in \mathbb{R}^k$ — скрытые представления признаков

Модель приближает линейную модель с квадратичными признаками, но для квадратичных признаков обучает не d^2, а kd весов, что позволяет избежать переобучения

Deep factorization machines

Идея: обучить общие эмбеддинги для факторизационной машины и глубокой нейронной сети



Graph-based методы

Построим двудольный граф, вершинами которого являются пользователи и объекты, а ребро между u и i существует, если пользователь u оценил объект i

В этом графе можно ввести похожесть между парой вершин $sim(x, y)$ как вероятность попасть в вершину y при случайном блуждании из вершины x

- Можно использовать похожесть между пользователем и объектом, чтобы сразу находить релевантные объекты
- Можно найти пользователей, похожих на данного, и рассматривать оцененные ими объекты
- Можно найти объекты, похожие на те, с которыми взаимодействовал пользователь

Graph-based методы

Можно сделать граф более сложным, добавив другие виды сущностей и связей — пользователей, фильмы, актеров и режиссеров.

Будем выписывать вершины в порядке их посещения при случайном блуждании. Данную последовательность можно рассматривать как текст и использовать стандартные решения для работы с текстами, такие как word2vec

Denoising Autoencoder

Идея: предположим, что для пользователя есть истинный вектор его оценок для всех объектов, а текущее состояние — это его зашумленная версия (некоторые значения пропущены)

На этапе обучения:

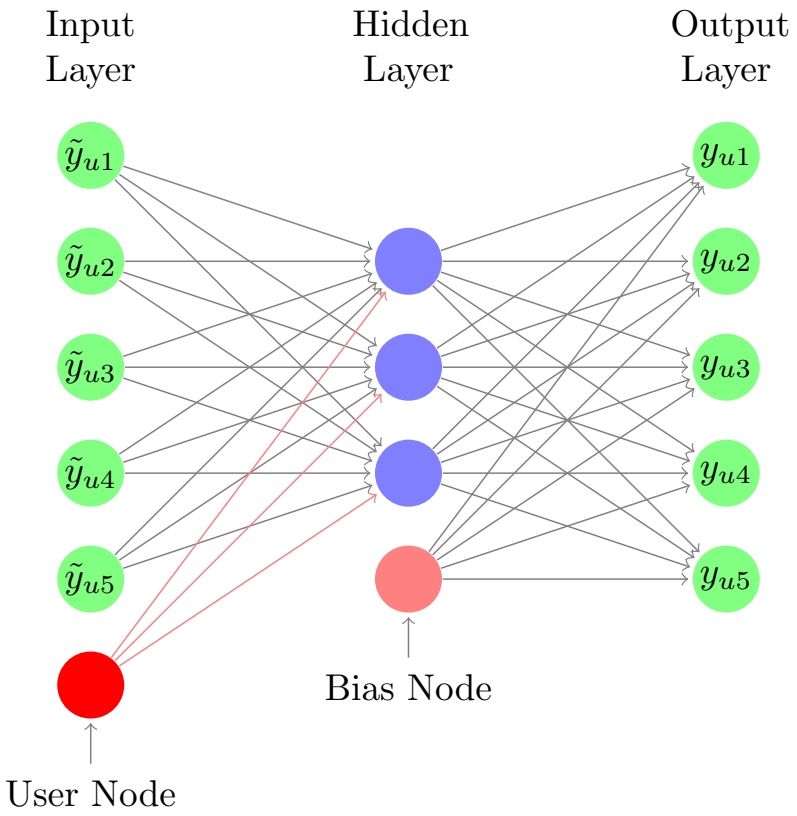
$y_{ui} = r_{ui}$

$\tilde{y}_{ui} = \delta(r_{ui}) =$

$$\begin{cases} r_{ui} \text{ с вероятностью } 1 - p \\ 0 \text{ с вероятностью } p \end{cases}$$

На этапе предсказания:

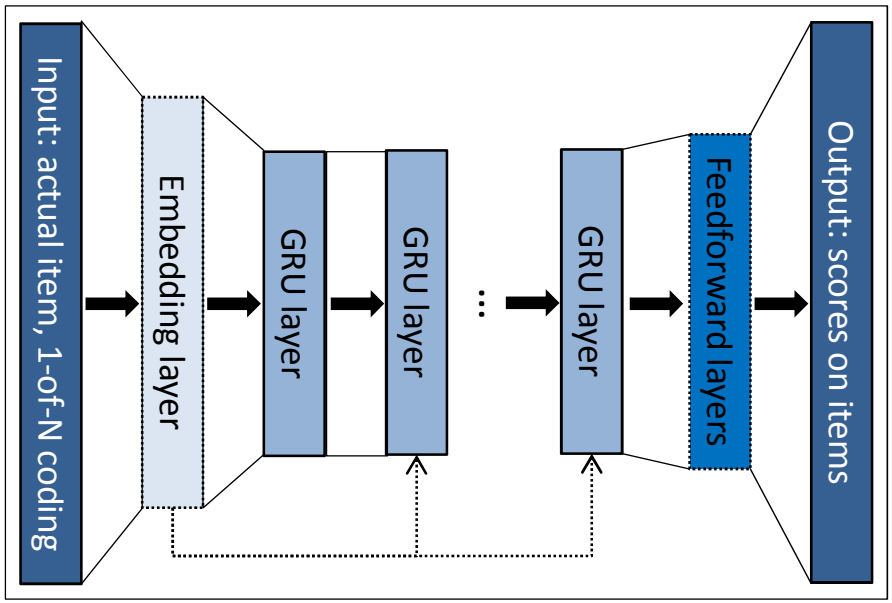
$\tilde{y}_{ui} = r_{ui}$



Рекомендации для сессии

Во многих сервисах (онлайн-радио, интернет-магазины) существует проблема рекомендаций внутри сессии — для нового пользователя нужно по первым нескольким его действиям предсказать дальнейшие и построить рекомендации

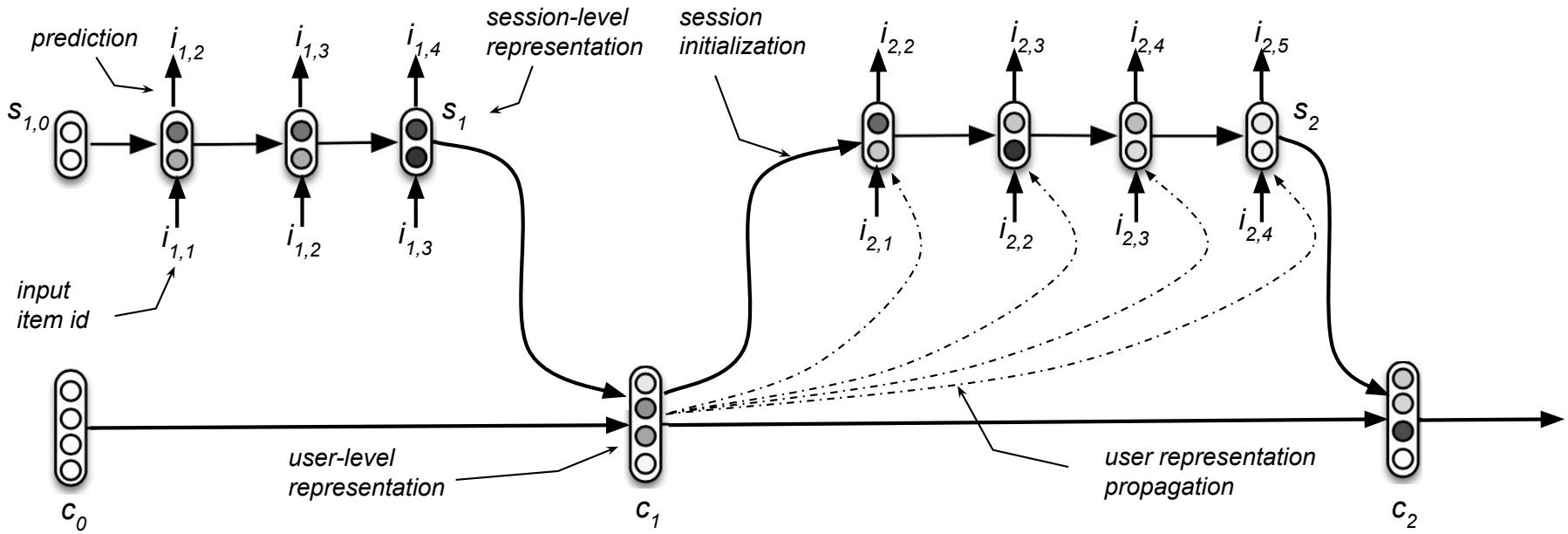
Идея: обучить рекуррентную нейронную сеть для предсказания следующего действия пользователя в сессии



Иерархические RNN

Иногда несмотря на то, что работа с сервисом разбита на сессии, все же есть информация о пользователе и его предыдущих сессиях

Можно сделать внешнюю рекуррентную сеть, которая будет использоваться для инициализации сессий и обновляться после каждой сессии



Что осталось за кадром

Субъективное качество рекомендаций определяется не только их релевантностью, на него также влияют следующие факторы:

- Новизна
  Не рекомендовать то, что пользователь уже видел (в том числе в рекомендациях)
- Serendipity — “Неочевидность”
  Рекомендовать не только песни того же исполнителя или фильмы с теми же актерами
- Разнообразие
  Как внутри одной рекомендации, так и между рекомендациями
  Не рекомендовать 10 песен одного исполнителя