Обзор рекомендательных систем

Коллаборативная фильтрация (→CF) использует историю прошлых действий юзеров. Имеющиеся взаимодействия, скорее всего, скоррелированы, что позволит восстановить пропущенные значения. Можно выделить 2 основных направления CF:

- ➔ *Memory-based methods (Neighborhood-based).* Значение рейтинга предсказывается на основе соседства. В целом соседство или близость может рассматриваться как эвристика. Соседство может быть между айтемами и между юзерами, поэтому методы делятся на 2 вида:
  - ◆ *User-based (user2user)*. Непроставленные рейтинги интересующего нас юзера предсказываются на основе рейтингов близких к нему юзеров. Необходимо определить, что будет считаться отражением соседства юзеров (метрика близости), и взвесить рейтинги юзеров-соседей.
  - ◆ *Item-based (item2item)*. Аналогично u2u-модели с той разницей, что в этом случае для таргет-айтема определяется множество близких айтемов. На основании того, какие рейтинги айтемам из этого множества поставил таргет-юзер, прогнозируется рейтинг таргет-айтема от таргет-юзера.

  Преимущество метода в интерпретируемости результатов. Недостаток в том, что матрица взаимодействий разрежена и может быть сложно найти соседей айтемов/юзеров, а также делать уверенные прогнозы.

- ➔ *Model-based methods*. Пространство взаимодействий юзеров и айтемов параметризуется и параметры обучаются в зависимости от подхода. Преимущество в том, что разреженность матрицы взаимодействий не мешает делать прогнозы рейтингов. Недостаток в том, что например, в случае матричной факторизации результаты едва ли интерпретируемые.

Content-Based (→CB) использует свойства айтемов, которые помогают отнести его к определённой к группе и на основании этого порекомендовать юзеру схожие айтемы. Так, для составления рекомендация для юзера u не нужны рейтинги других юзеров: лишь рейтинги юзера u и описания товаров. За счёт того, что близость ищется не (только) по рейтингам, CB алгоритмы позволяют бороться с проблемой холодного старта. К недостаткам CB относят следующее: (1) рекомендации могут быть очевидными; (2) позволяя бороться с проблемой холодного старта для айтема, методы CB плохо решают такую проблему для юзера, и по-прежнему требуют большего количества рейтингов от юзера.

Knowledge-Based (→KB) применяется к айтемам, с которыми юзеры взаимодействуют в целом редко (или мало в случае новых айтемов), при времени между взаимодействиями достаточно, чтобы предпочтения юзера могли поменяться или в айтем были внесены изменения. Так, становится сложно пронозировать рейтинг по прошлым взаимодействиям. В этом случае не используются исторические данные о юзере, а собираются знания о его предпочтениях, которые юзер должен явно выразить. По типу получаемого знания выделяют 2 направления KB:

- ➔ *Constraint-based*. Юзером задаются ограничения по отношению к свойствам айтема (рамки). Появляются правила отсеивания айтемов по принципу соответствия желаниям юзера. Юзер меняет ограничения, пока выдача его не удовлетворит, т.е. итартивно.
- ➔ *Case-based*. Юзер задаёт идеально подходящий айтем. Ему рекомендуются близкие по свойствам к идеальному айтемы.

KB схожа с CB в том смысле, что основана на свойствах товаров, и поэтому перенимает недостатки CB-методов: рекомендации могут быть очевидными, однако помогает решать проблему холодного старта. Разница в том, что CB также использует взаимодействия юзера в прошлом, а в случае KB юзер сам указывает свои интересы.

Hybrid and Ensemble-Based (→HB) объединяет результаты нескольких рекомендательных систем (не всегда в единый). Детальное разделение выделяет 7 категорий HB:

- ➔ *Wighted*. Отдельно вычисляются прогнозы рейтингов q алгоритмов. Алгоритм i заполняет пропуски в исходной матрице R, и получается матрица $\widehat{R_i}$. Итоговая матрица $\widehat{R}$ вычисляется как взвешенное среднее прогнозов q алгоритмов: $\widehat{R} = \sum_{i=1}^{q} \alpha_i \widehat{R_i}$. Для подбора весов $\bar{\alpha} = (\alpha_1,\dots,\alpha_q)$ есть используется следующая процедура.

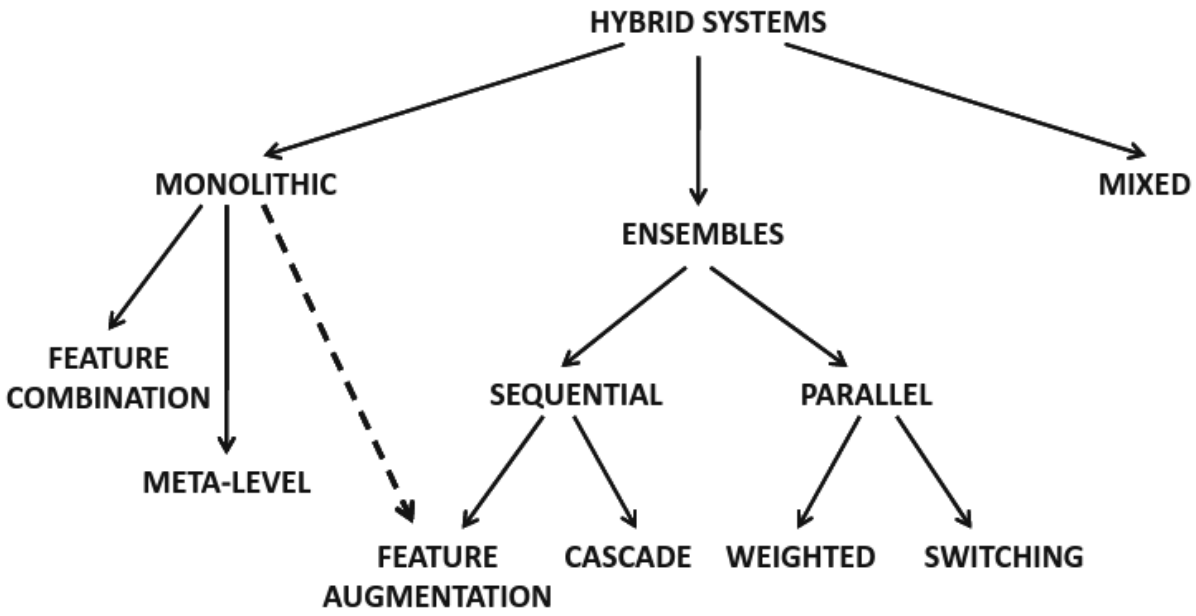
  Выборка делится на train и test части, последнюю обозначим за H. В качестве целевой функции используется MAE: $MAE(\bar{\alpha}) = \frac{\sum_{(u,j)\in H} |\hat{r}_{uj} - r_{uj}|}{|H|}$. В целом подход является аналогом линейной регрессии, и чтобы результаты были более устойчивыми к шумам и выбросам вместо MSE используется MAE. В начале всем весам присваиваются одинаковые значения: $\alpha_1 = \dots = \alpha_q = \frac{1}{q}$. Затем на H считается градиент: $\frac{\partial MAE(\bar{\alpha})}{\partial \alpha_i} = \frac{\sum_{(u,j)\in H} \frac{\partial |\hat{r}_{uj} - r_{uj}|}{\partial \alpha_i}}{|H|} = \frac{\sum_{(u,j)\in H} sign(\hat{r}_{uj} - r_{uj}) \cdot \hat{r}_{uj}}{|H|}$, $\nabla_{\bar{\alpha}} MAE = (\frac{\partial MAE(\bar{\alpha})}{\partial \alpha_1}, \dots, \frac{\partial MAE(\bar{\alpha})}{\partial \alpha_q})$. Затем итеративно вектор весов $\bar{\alpha}$ обновляется значением антиградиента MAE: $\bar{\alpha}^{(t+1)} = \bar{\alpha}^{(t)} - \gamma \cdot \nabla_{\bar{\alpha}} MAE$. Опционально можно добавлять регуляризацию.

- ➔ *Switching*. Во время холодного старта используется более подходящий для него алгоритм, а потом он проседает по качеству, поэтому после прогрева юзера переключаются на другой алгоритм, отсюда и название. Например, использовать алгоритмы из KB или CB на первом шаге, а затем применять алгоритмы CF.
  Другими словами, на каждом шаге используются алгоритмы, сильнее роняющие ошибку (MSE/MAE). Как в weighted, выбираться модели могут по делением на train/test (hold-out).
- ➔ *Cascade*. Каждый последующий алгоритм учитывает результат, выданный предыдущим. Выделяется 2 категории каскадных гибридов:

    - *Successive Refinement of Recommendations*.
    - *Boosting*. Более подробно ниже.
- ➔ *Feature augmentation*. Результат работы первого алгоритма подаётся на вход второму в качестве фичей. Сперва CB-алгоритм заполняет разреженную матрицу рейтингов псевдорейтингами. Далее прогнозы псевдорейтингов корректируются CF-алгоритмов. Итоговый прогноз собирается как взвешенное среднее между прогнозом CB и CF.
- ➔ *Feature combination*. В оригинальной работе предлагается рассматривать как задачу классификации (like/dislike) на примере фильмов. Сначала собираются признаки 3 типов:
    - Коллаборативные фичи: любимые фильмы юзера u и юзеры, которым нравится фильм i. “Нравится” является эвристикой: фильм нравится юзеру u, если рейтинг, поставленный им фильму i, больше определённого порогового квантиля (70, 75, 80 и т.д.). Получается кортеж из двух множеств и рейтинга вида:
      $(\{\text{множество фильмов, нравящихся юзеру}\}, \{\text{множество юзеров, которым нравится фильм}$
      $\text{нравится/не нравится фильм } i \text{ юзеру } u \text{ согласно квантильному порогу})$;
    - Контентные фичи: метаданные о фильме (жанр, длительность и т.д.);
    - Гибридные фичи: юзеры, которым нравится определённый жанр фильма.

  На основании этого определяется вероятность того, что айтем юзеру понравится.
- ➔ *Meta-level*. На первом этапе происходит подготовка данных для второго, а не выдача из рекомендованных первым алгоритмом айтемов. Например, запускается CB-алгоритм, он находит айтемы, близкие к предпочтениям юзеров. Из них извлекаются описательные признаки (слова из описания айтема) и каждый из таких признаков получает веса согласно близости айтемов. Это в том числе отбросит нерелевантные айтемы. Затем запускается CF-алгоритм, который находит юзеров-соседей таргет-юзера в новом признаковом пространстве. Его пропущенные рейтинги заполняются как взвешенное среднее рейтингов из этой группы.
  В другом варианте алгоритм может сразу искать юзеров-соседей по их характеристикам в профиле.
- ➔ *Mixed*. Выдачи разных алгоритмов показываются юзеру рядом друг с другом. Применяется для айтемов, одновременно обладающих разнородными свойствами.

Подход к классификации позаимствован из учебника.



Архитектура решения

На первом этапе будут отбираться кандидаты в ранжирование алгоритмами, основанные на CF. На втором этапе для ранжирования будет применяться градиентный бустинг. Видимо, такую архитектуру можно отнести к каскадному типу гибридных рекомендательных систем.

Ансамбль в задаче ранжирования (построения рекомендаций) имеет аналогичные свойства с ансамблем в задаче классификации. В частности, выполняется принцип Bias Variance Decomposition, и в *работе* на примере показывается, что агрегирование моделей с помощью коллаборативной фильтрации ведёт к улучшению качества.

Этап I. Генерирование кандидатов

Алгоритмы можно поделить на и. В качестве таких моделей будут использованы:

- ➔ *Матричная факторизация [CF].*
- ➔ *BM-25 Recommender [CB].* Является модификацией или более общим случаем (как посмотреть) TF-IDF, т.е. item2item модель. Используется в качестве меры близости текста запроса и текста документа. В нашем случае документ – это айтем, запрос – это юзер, тексты – это юзер-айтем взаимодействия. Формула для вычисления скора айтема i будет примерно следующего вида:

$$BM25(i) = \sum_{u=1}^{|U|} \frac{r_{ui} \cdot (K_1+1) \cdot IDF(u)}{r_{ui} + K_1 \cdot (1 - B + B \cdot \frac{\sum_{u=1}^{|U|} r_{ui}}{\frac{\sum_{i=1}^{|I|}\sum_{u=1}^{|U|} r_{ui}}{|I|}})} = \sum_{u=1}^{|U|} \frac{r_{ui} \cdot (K_1+1) \cdot \frac{|I|}{1+\sum_{i=1}^{|I|} p_{ui}}}{r_{ui} + K_1 \cdot (1 - B + B \cdot \frac{\sum_{u=1}^{|U|} r_{ui}}{\frac{\sum_{i=1}^{|I|}\sum_{u=1}^{|U|} r_{ui}}{|I|}})}$$

где $\sum_{i=1}^{|I|} p_{ui}$ - какие товары покупал юзер u, p_{ui} – это бинаризация r_{ui}

Этап II. Ранжирование кандидатов

Переложения ансамбля с задачи классификации на задачу ранжирования рекомендаций было описано в работе *Boosting Simple Collaborative Filtering Models Using Ensemble Methods*.

Как библиотека для бустинга, вероятно, будет использоваться LightGBM. Во-первых, LightGBM, как правило, отрабатывает быстрее других библиотек. Во-вторых, можно использовать LambdaRank. Переход от RankNet изложен в работе *From RankNet to LambdaRank to LambdaMART: An Overview*.

Лучше понять рассуждения оригинальной работы Microsoft мне помогла статья The inner workings of the lambdarank objective in LightGBM. Изложу идею своими словами.

Необходимо как-то оценить качество ранжирования. Базовая идея (в контексте pairwise подхода) – считать долю дефектных пар (→ДДП). Для краткости обозначим число пар за n. Будем считать, что айтем i релевантнее айтема j. Тогда

$ДДП = \frac{1}{n} \cdot \sum_{(i,j)\in R} [a(x_i) - a(x_j) < 0]$. Такая функция из индикаторов не дифференцируема. Аналогично задаче классификации, ограничим ДДП непрерывной дифференцируемой функций сверху – LogLoss-ом, но для случая ранжирования. Для этого от функции отступа как разности констант перейдём к задаче оптимизации правдоподобия. Для удобства берут логарифм правдоподобия, а вероятность правильной и неправильной пар представляют функция сигмоиды. Негативные пары симметричны позитивным, поэтому в логлоссе слагаемое, отвечающее за штраф для негативных пар отбрасывается. Получаем функцию вида:

$lnL = \sum_{(i,j)\in R} ln(1 + exp(-\sigma(a(x_i) - a(x_j))))$ - это logistic pairwise loss, RankNet.

О LambdaRank корректно говорить как о преобразовании градиента, чем как о целевой функции, поэтому выпишем градиент: $\frac{\partial lnL_{ij}}{\partial a(x_i)} = \frac{-\sigma \cdot exp(-\sigma(a(x_j)-a(x_i)))}{1+exp(-\sigma(a(x_i)-a(x_j)))}$.

Тогда шаг градиентного спуска для обновления весов будет:

$w^{(k)} = w^{(k-1)} + \eta^{(k)} \cdot \frac{\sigma e^{-\sigma(a(x_j)-a(x_i))}}{1+e^{-\sigma(a(x_j)-a(x_i))}}$.

Можно показать, что из ДДП выводится ROC-AUC, а ROC-AUC отражает, насколько хорошо отсортирована вся выдача. Нам же важно сделать как можно более релевантной именно верх выдачи. Показателем этого является nDCG. Делается костыль.

Logistic pairwise loss домножается на то, на сколько меняется значение nDCG от перестановки айтемов i и j местами, обозначаемое за $|\Delta NDCG_{ij}|$. Если поменять местами объекты внизу списка, то значение будет ≈ 0, а если в начале, то большим. Так модель будет подстраиваться под верх выдачи. Тогда шаг градиентного спуска будет таким: $w^{(k)} = w^{(k-1)} + \eta^{(k)} \cdot \frac{\sigma e^{-\sigma(a(x_j)-a(x_i))} \cdot |\Delta NDCG_{ij}|}{1+e^{-\sigma(a(x_j)-a(x_i))}}$. Этот подход называется LambdaRank.

Перед началом отбора кандидатов можно попробовать применить фильтр, отрбрасывающий товары, которые являются сезонными и сейчас не их сезон. Есть айтемы, продажи которых зимой намного выше, чем весной, и, вероятно, не стоит даже искать среди товаров, актуальных зимой, кандидаты в ранжирование.