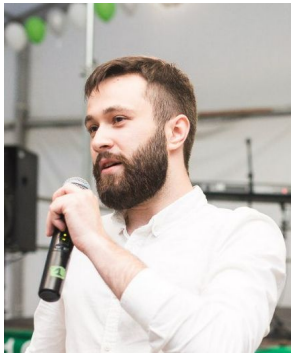
# Your First RecSys

**Даниил Потапов**
Руководитель группы персонализации и рекомендательных систем
МТС BigData



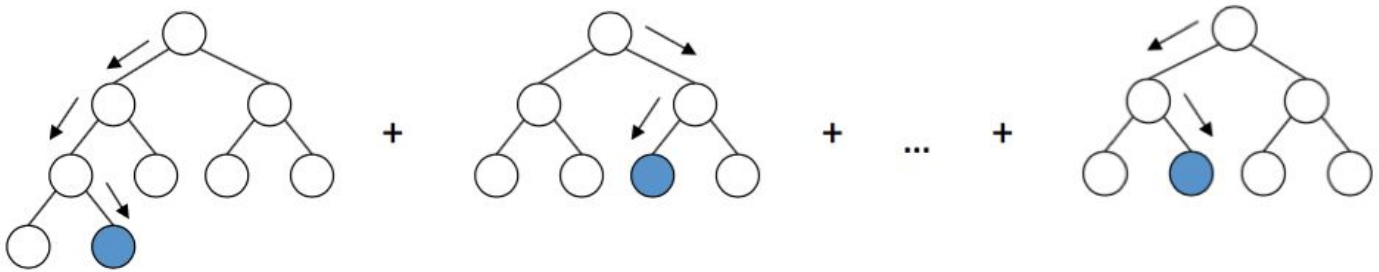
# GBDT алгоритмы для рекомендательных систем

# План лекции

- GBDT
- Постановка задачи ранжирования
- Двухэтапная архитектура
- CatBoost

# GBDT

GBDT - Gradient Boosting on Decision Trees



credit

# GBDT

- XGBoost
- LightGBM
- CatBoost
- Scikit-learn
  - sklearn.ensemble.GradientBoostingClassifier/Regressor
  - sklearn.ensemble.HistGradientBoostingClassifier/Regressor

МТС

# Задача ранжирования

**Регрессия**

Мы хотим найти такую $f(x)$, которая для $x$ предскажет $y \in R$.

**Классификация**

Мы хотим найти такую $f(x)$, которая для $x$ предскажет класс $y \in \{1,2,...,N\}$

# Задача ранжирования

**Ранжирование**

Мы хотим найти такую *f(u, Items)*, которая для заданного пользователя *u* и набора объектов *Items={i1,i2,...,iM}* предскажет ранг для каждого объекта в наборе

# Задача ранжирования

From RankNet to LambdaRank to LambdaMART: An Overview Christopher J.C. Burges

МТС

# Задача ранжирования

**Связь с регрессией**

$f(u, i\_K) = y\_K \in R$ - рейтинг, время просмотра и тд

Ранжируем $\{i\_1,i\_2,...,i\_M\}$ по соответствующим $\{y\_1,y\_2,...,y\_M\}$

**Связь с бинарной классификацией**

$f(u, i\_K) = p\_K \in \{0, 1\}$ - вероятность клика или покупки

Ранжируем $\{i\_1,i\_2,...,i\_M\}$ по соответствующим $\{p\_1,p\_2,...,p\_M\}$

# Задача ранжирования

Основные подходы:

- Pointwise
- Pairwise
- Listwise

# Задача ранжирования. Pointwise

**Основная идея** - свести задачу от *f(u, Items)* к *f(u, i_K)*

По факту получается либо задача регрессии, либо задача классификации

# Задача ранжирования. Pairwise

**Основная идея** - также, как и в Pointwise, обучаем $f(u, i\_K)$.

Но для обучения уже используем два объекта $(u, i\_M, i\_N)$ и бинарный таргет, $f(u, i\_M) > f(u, i\_N)$

# Задача ранжирования. Listwise

**Основная идея** - использовать информацию о целом наборе *Items* для обучения.

Получается из Pairwise, если мы считаем ошибку суммарно по парам в рамках одного запроса пользователя

# Двухэтапная архитектура

- Генерация кандидатов более простыми моделями
  - Популярное
  - Item2item, User2user
  - ALS
  - LightFM
  - Random
- Реранжирование списка кандидатов с фичами более тяжелой моделью
  - GBDT
  - Neural nets

# Двухэтапная архитектура. Таргет

**Позитивные примеры**

- Известные взаимодействия
- Порог по рейтингу

**Негативные примеры**

- Известные взаимодействия с негативной оценкой
  - Порог по рейтингу
  - Возврат после покупки
- Случайные
- На основе простых моделей

# Двухэтапная архитектура. Таргет

Цель сэмплирования негативных примеров - научить модель выделять именно те зависимости, которые привели к позитивному взаимодействию.

Поэтому лучше использовать генерацию на основе простых моделей и с привязкой по времени.

Качество негативного сэмплирования - качество простых моделей на заданном топ-N.

# Двухэтапная архитектура. Валидация

Два подхода:

- Случайное разбиение
  - По группе таргета из одного взаимодействия (positive + N negatives)
- По времени
  - Имитируем процесс эксплуатации в проде

# Двухэтапная архитектура. Кандидаты

Популярное:

- Легко посчитать вплоть по каждому дню с разным окном
- Хранить можно в таблице вида [item_id, date, popularity, rank]

kNN и матричные разложения

- Накладно хранить много моделей, поэтому можно увеличивать “временной шаг”, 1 модель на месяц

Варианты ансамблирования:

- Rank average
- Model priority

# Двухэтапная архитектура. Кандидаты

Для train после построения кандидатов надо убрать дубликаты.

Для test мы вначале строим кандидатов по всем пользователям из test, а потом к ним присоединяем кандидатов через left join.

На этом этапе мы сразу можем оценить качество кандидатов.

# Двухэтапная архитектура. Кандидаты

| | user_id | item_id | date |
|---|---|---|---|
| 0 | 1 | 1 | 2020-03-02 |
| 1 | 1 | 2 | 2020-05-17 |
| 2 | 1 | 3 | 2020-06-23 |

| | user_id | item_id | date | target |
|---|---|---|---|---|
| 0 | 1 | 1 | 2020-03-02 | 1 |
| 1 | 1 | 3 | 2020-03-02 | 0 |
| 2 | 1 | 5 | 2020-03-02 | 0 |
| 3 | 1 | 6 | 2020-03-02 | 0 |
| 4 | 1 | 7 | 2020-03-02 | 0 |
| 5 | 1 | 2 | 2020-03-02 | 0 |
| 6 | 1 | 2 | 2020-05-17 | 1 |
| 7 | 1 | 4 | 2020-05-17 | 0 |
| 8 | 1 | 2 | 2020-05-17 | 1 |
| 9 | 1 | 6 | 2020-05-17 | 0 |
| 10 | 1 | 5 | 2020-05-17 | 0 |
| 11 | 1 | 8 | 2020-05-17 | 0 |

| | user_id | item_id | date | target |
|---|---|---|---|---|
| 0 | 1 | 1 | 2020-03-02 | 1 |
| 1 | 1 | 3 | 2020-03-02 | 0 |
| 2 | 1 | 5 | 2020-03-02 | 0 |
| 3 | 1 | 6 | 2020-03-02 | 0 |
| 4 | 1 | 7 | 2020-03-02 | 0 |
| 5 | 1 | 2 | 2020-03-02 | 0 |
| 6 | 1 | 2 | 2020-05-17 | 1 |
| 7 | 1 | 4 | 2020-05-17 | 0 |
| 9 | 1 | 6 | 2020-05-17 | 0 |
| 10 | 1 | 5 | 2020-05-17 | 0 |
| 11 | 1 | 8 | 2020-05-17 | 0 |

# Двухэтапная архитектура. Фичи

После генерации кандидатов получается каркас нашего датасета:

- user_id
- item_id
- date
- target

На этот каркас через join’ы присоединяются фичи. На практике это самый тяжелый этап.

# CatBoost

Objectives:

- RMSE
- QueryRMSE
- PairLogit
- PairLogitPairwise
- YetiRank
- YetiRankPairwise

Metrics:

- Precision@K, Recall@K
- MAP
- DCG, NDCG
- PFound

https://catboost.ai/docs/concepts/loss-functions-ranking.html

# CatBoost

catboost.Pool - класс для более оптимального использования датасета во время обучения

Ключевые аргументы:

- data - наша таблица
- label - таргет
- group_id - user_id
- pairs - матрица [Nx2] с парами для Pairwise

# CatBoost

```
catboost_model = catboost.CatBoost({
    'loss_function': 'YetiRank',
    'learning_rate': 0.1,
    'iterations': 1000,
    'early_stopping_rounds': 30,
    'custom_metric': ['RecallAt:top=1', 'RecallAt:top=10', 'PrecisionAt:top=10'],
    'eval_metric': 'MAP:top=3',
})


catboost_model.fit(
    catboost_pool_train,
    eval_set=catboost_pool_test,
    logging_level='Silent',
    plot=True
)
```

# LightGBM

Objectives:

- LambdaRank
- XE_NDCG_MART

Metrics:

- MAP
- NDCG

https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRanker.html

# LightGBM

```
lgbm_model = lightgbm.LGBMRanker()

lgbm_model.fit(
        X_train,
        y_train,
        group=users_train,
        eval_set=[(X_test, y_test)],
        eval_group=[users_test],
        eval_at=[1, 2, 5, 10],
        early_stopping_rounds=50)
```

Всем спасибо за внимание :)

Вопросы можно задавать здесь

 Telegram чате курса

 sharthZ23