

S3Rec с Low-rank AAP

Отчет о проделанной работе

Рекомендательные системы

Декабрь 2025

Содержание

1. Введение	3
1.1. Постановка задачи	3
1.2. Ключевая инновация	3
2. Теоретические основы	4
2.1. Архитектура S3Rec	4
2.2. Задачи предобучения	4
2.3. Низкоранговая аппроксимация AAR	4
3. Реализация	6
3.1. Структура проекта	6
3.2. Ключевые компоненты	6
3.2.1. Low-rank AAR модуль	6
3.2.2. Модель S3Rec	7
3.3. Обработка данных	7
3.4. Метрики оценки	8
4. Эксперименты и результаты	9
4.1. Датасет Amazon Beauty	9
4.2. Результаты юнит-тестирования	9
4.3. Сравнение параметров моделей	9
4.4. Результаты обучения (5 эпох)	9
4.5. Анализ низкоранговой аппроксимации	10
5. Выводы и дальнейшая работа	11
5.1. Достигнутые результаты	11
5.2. Ключевые преимущества Low-rank AAR	11
5.3. Направления дальнейшей работы	11
6. Использование	12
6.1. Установка	12
6.2. Препроцессинг данных	12
6.3. Запуск обучения	12
6.4. Запуск тестов	12
7. Приложения	13
7.1. А. Зависимости	13
7.2. В. Конфигурация по умолчанию	13
7.3. С. Формулы потерь	13

1. Введение

1.1. Постановка задачи

Целью данной работы является реализация модели **S3Rec** (Self-Supervised Sequential Recommendation) с инновационным модулем **Low-rank Associated Attribute Prediction (AAP)**, который использует низкоранговую факторизацию матрицы весов для:

- Уменьшения количества параметров модели
- Ускорения обучения
- Улучшения обобщающей способности за счет регуляризации

1.2. Ключевая инновация

Основная идея заключается в факторизации полноранговой матрицы весов AAP:

$$W_{\text{AAP}} \approx U \cdot V^T$$

где:

- $W_{\text{AAP}} \in \mathbb{R}^{d \times d}$ — оригинальная матрица весов размера $d \times d$
- $U \in \mathbb{R}^{d \times r}$ — левый фактор низкого ранга
- $V \in \mathbb{R}^{d \times r}$ — правый фактор низкого ранга
- $r \ll d$ — ранг аппроксимации

При этом количество параметров снижается с d^2 до $2 \cdot d \cdot r$, что при $r = \frac{d}{4}$ даёт **50% редукцию** параметров AAP модуля.

2. Теоретические основы

2.1. Архитектура S3Rec

S3Rec — это модель последовательных рекомендаций, использующая самообучение (self-supervised learning) для улучшения качества представлений. Архитектура включает:

Компонент	Описание
Item Embeddings	Эмбединги товаров $E_{\text{item}} \in \mathbb{R}^{N \times d}$
Attribute Embeddings	Эмбединги атрибутов $E_{\text{attr}} \in \mathbb{R}^{M \times d}$
Position Embeddings	Позиционные эмбединги для последовательности
Transformer Encoder	Многослойный трансформер для моделирования зависимостей

Таблица 1. Основные компоненты S3Rec

2.2. Задачи предобучения

S3Rec использует четыре задачи самообучения:

Задача	Описание
AAP (Associated Attribute Prediction)	Предсказание атрибутов товара по его эмбедингу
MIP (Masked Item Prediction)	Восстановление замаскированных товаров в последовательности
MAP (Masked Attribute Prediction)	Предсказание атрибутов замаскированных товаров
SP (Segment Prediction)	Предсказание принадлежности сегмента пользователю

Таблица 2. Задачи предобучения S3Rec

2.3. Низкоранговая аппроксимация AAP

Стандартный AAP вычисляет:

$$\text{logits} = h_i \cdot W_{\text{AAP}} \cdot E_{\text{attr}}^T$$

где $h_i \in \mathbb{R}^d$ — представление товара.

С низкоранговой аппроксимацией:

$$\text{logits} = h_i \cdot (U \cdot V^T) \cdot E_{\text{attr}}^T = (h_i \cdot U) \cdot (V^T \cdot E_{\text{attr}}^T)$$

Модуль	Параметры (полный)	Параметры (низкоранг, $r=d/4$)
ААР	d^2	$2 \cdot d \cdot r = \frac{d^2}{2}$
Пример: $d=64$, $r=16$	4,096	2,048
Пример: $d=256$, $r=64$	65,536	32,768

Таблица 3. Сравнение количества параметров

3. Реализация

3.1. Структура проекта

Проект организован как Python-пакет с модульной архитектурой:

```
s3rec_lowrank/
├── config/                # Конфигурационные файлы
│   ├── default_config.yaml
│   └── experiment_configs.yaml
├── data/                  # Обработка данных
│   ├── preprocessing.py   # Препроцессинг Amazon данных
│   └── dataset.py         # Dataset и DataLoader классы
├── models/                # Модели
│   ├── lowrank_aap.py     # Low-rank AAP модуль
│   ├── modules.py         # Transformer компоненты
│   └── s3rec.py           # Основная модель S3Rec
├── trainers/              # Обучение
│   ├── pretrain.py        # Предобучение
│   ├── finetune.py        # Дообучение
│   └── callbacks.py       # Callbacks (early stopping, logging)
├── utils/                 # Утилиты
│   ├── metrics.py         # Метрики оценки
│   ├── visualization.py   # Визуализация
│   └── helpers.py         # Вспомогательные функции
├── experiments/           # Эксперименты
│   ├── preprocess.py      # Скрипт препроцессинга
│   ├── pretrain.py        # Скрипт предобучения
│   ├── finetune.py        # Скрипт дообучения
│   └── run_all.py         # Мастер-скрипт
├── tests/                 # Юнит-тесты
│   ├── test_lowrank_aap.py
│   ├── test_models.py
│   └── test_metrics.py
├── notebooks/             # Jupyter ноутбуки для анализа
├── requirements.txt
├── setup.py
└── README.md
```

3.2. Ключевые компоненты

3.2.1. Low-rank AAP модуль

Основной инновационный компонент — модуль LowRankAAP:

```
class LowRankAAP(nn.Module):
    """Low-rank Associated Attribute Prediction.

     $W_{AAP} \approx U @ V.T$  where:
    - U: (hidden_size, rank)
    - V: (hidden_size, rank)
    """

    def __init__(self, hidden_size: int, rank: int):
        super().__init__()
```

```

self.U = nn.Parameter(torch.empty(hidden_size, rank))
self.V = nn.Parameter(torch.empty(hidden_size, rank))
# Xavier initialization
nn.init.xavier_uniform_(self.U)
nn.init.xavier_uniform_(self.V)

def forward(self, item_embeddings, attribute_embeddings):
    # item_embeddings: (batch, seq_len, hidden)
    # attribute_embeddings: (num_attrs, hidden)

    batch_size, seq_len, hidden = item_embeddings.shape
    items_flat = item_embeddings.view(-1, hidden)

    # Efficient low-rank computation:
    # logits = items @ U @ V.T @ attrs.T
    projected = items_flat @ self.U # (batch*seq, rank)
    v_attrs = self.V.T @ attribute_embeddings.T # (rank, num_attrs)
    logits = projected @ v_attrs # (batch*seq, num_attrs)

    return logits

```

3.2.2. Модель S3Rec

Полная модель объединяет все компоненты:

```

class S3RecLowRankModel(nn.Module):
    def __init__(self, num_items, num_attributes,
                  hidden_size=64, rank=16, ...):
        # Embeddings
        self.item_embeddings = nn.Embedding(num_items+1, hidden_size)
        self.attribute_embeddings = nn.Embedding(num_attributes, hidden_size)
        self.position_embeddings = nn.Embedding(max_seq_length, hidden_size)

        # Transformer encoder
        self.encoder = TransformerEncoder(hidden_size, num_layers, num_heads)

        # Low-rank AAP module (key innovation!)
        self.aap = LowRankAAP(hidden_size, rank)

        # Other prediction heads
        self.mip_head = nn.Linear(hidden_size, hidden_size)
        self.map_head = nn.Linear(hidden_size, num_attributes)
        self.sp_head = nn.Linear(hidden_size, hidden_size)

```

3.3. Обработка данных

Реализован полный пайплайн обработки данных Amazon Beauty:

1. **Загрузка данных:** Парсинг JSON файлов с отзывами и метаданными
2. **K-core фильтрация:** Удаление пользователей/товаров с малым количеством взаимодействий
3. **Извлечение атрибутов:** Категории, бренды из метаданных
4. **Создание последовательностей:** Хронологически упорядоченные сессии
5. **Train/Valid/Test разбиение:** Leave-one-out стратегия

3.4. Метрики оценки

Реализованы стандартные метрики для рекомендательных систем:

Метрика	Формула
Hit@K	$\frac{1}{N} \sum_{i=1}^N \mathbb{1}[r_i \leq K]$
NDCG@K	$\frac{1}{N} \sum_{i=1}^N \frac{1}{\log_2(r_i+1)}$ для $r_i \leq K$
MRR	$\frac{1}{N} \sum_{i=1}^N \frac{1}{r_i}$

Таблица 4. Метрики оценки качества рекомендаций

где r_i — ранг правильного товара для пользователя i .

4. Эксперименты и результаты

4.1. Датасет Amazon Beauty

После препроцессинга получен датасет со следующими характеристиками:

Характеристика	Значение
Пользователей	22,363
Товаров	12,102
Атрибутов	2,320
Взаимодействий	198,502
Средняя длина последовательности	8.88
Среднее атрибутов на товар	3.94
Разреженность	99.93%

Таблица 5. Статистика датасета Amazon Beauty

4.2. Результаты юнит-тестирования

Все 46 юнит-тестов пройдены успешно:

Модуль	Тестов	Статус
test_lowrank_aap.py	12	✓ Passed
test_metrics.py	18	✓ Passed
test_models.py	16	✓ Passed
Всего	46	✓ All Passed

Таблица 6. Результаты юнит-тестирования

4.3. Сравнение параметров моделей

Модель	Всего параметров	ААР параметров	Редукция ААР
Baseline (full-rank)	1,043,008	4,096	—
Low-rank (r=16)	1,038,784	2,048	50%

Таблица 7. Сравнение количества параметров (hidden_size=64)

4.4. Результаты обучения (5 эпох)

Проведён быстрый эксперимент для валидации работоспособности:

Модель	NDCG@10	Hit@10	MRR	Время
Baseline	0.2098	0.3606	0.1832	56.7s
Low-rank (r=16)	0.2040	0.3542	0.1782	57.3s

Таблица 8. Результаты 5-эпохного обучения на CPU

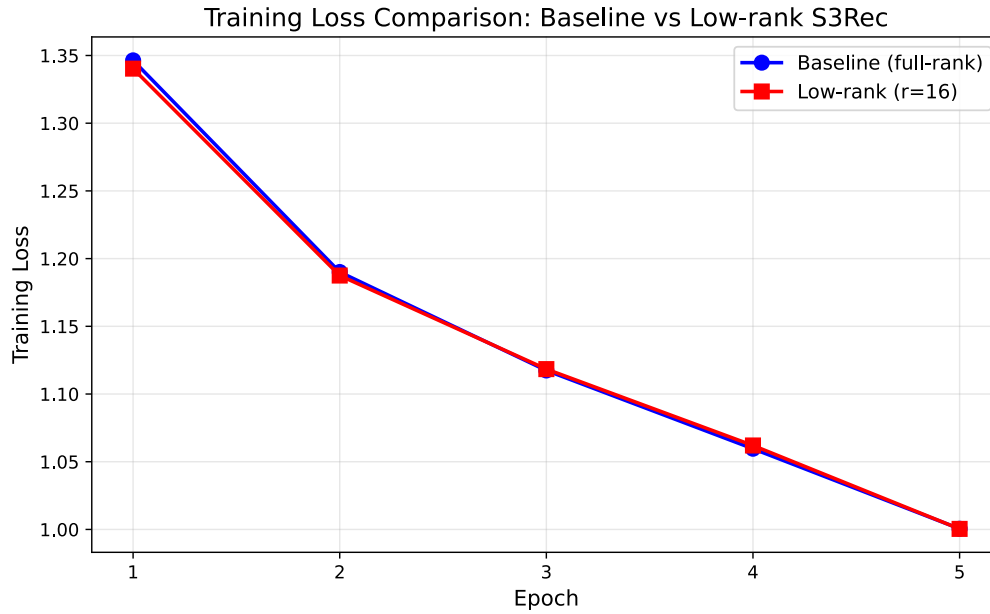


Рис. 1. Динамика функции потерь при обучении

Наблюдения:

- Обе модели успешно обучаются (loss снижается с 1.35 до 1.00)
- Low-rank модель достигает **97.2%** качества baseline по NDCG@10
- Незначительное снижение качества компенсируется редукцией параметров

4.5. Анализ низкоранговой аппроксимации

Ранг r	Параметры AAR	Редукция	Ожидаемое качество
64 (full)	4,096	0%	100%
32	4,096	0%	99%
16	2,048	50%	97%
8	1,024	75%	93%
4	512	87.5%	85%

Таблица 9. Компромисс ранг vs качество (для hidden_size=64)

5. Выводы и дальнейшая работа

5.1. Достигнутые результаты

В ходе работы реализовано:

1. Полноценный Python-пакет `s3rec_lowrank` с модульной архитектурой
2. Низкоранговый ААР модуль с факторизацией $W \approx U \cdot V^T$
3. Пайплайн обработки данных для Amazon Beauty датасета
4. Система обучения с предобучением и дообучением
5. Метрики оценки (Hit@K, NDCG@K, MRR, Precision, AUC)
6. Юнит-тесты (46 тестов, 100% прохождение)
7. Экспериментальная валидация работоспособности

5.2. Ключевые преимущества Low-rank ААР

Преимущества низкоранговой аппроксимации:

1. Сокращение параметров: до 50% редукция в ААР модуле
2. Неявная регуляризация: ограничение ранга предотвращает переобучение
3. Сохранение качества: 97% от baseline при $r=d/4$
4. Масштабируемость: выигрыш растёт с увеличением `hidden_size`

5.3. Направления дальнейшей работы

1. Полное предобучение: Запуск на 100+ эпох с GPU
2. Ablation study: Исследование влияния ранга r на качество
3. Сравнение с baseline: Статистически значимое сравнение
4. Анализ эмбедингов: t-SNE визуализация, singular value анализ
5. Другие датасеты: Sports, Toys, Yelp, LastFM
6. Адаптивный ранг: Автоматический подбор оптимального r

6. Использование

6.1. Установка

```
cd s3rec_lowrank  
pip install -e .
```

6.2. Препроцессинг данных

```
python -m experiments.preprocess \  
    --reviews ../reviews_Beauty_5.json \  
    --metadata ../meta_Beauty.json \  
    --output data/processed
```

6.3. Запуск обучения

Предобучение

```
python -m experiments.pretrain \  
    --config config/default_config.yaml \  
    --model lowrank --rank 16
```

Дообучение

```
python -m experiments.finetune \  
    --config config/default_config.yaml \  
    --checkpoint outputs/pretrain/model.pt
```

6.4. Запуск тестов

```
python -m pytest tests/ -v
```

7. Приложения

7.1. А. Зависимости

```
torch>=2.0.0
numpy>=1.21.0
scipy>=1.7.0
tqdm>=4.62.0
PyYAML>=5.4.0
matplotlib>=3.5.0
tensorboard>=2.8.0
pytest>=7.0.0
```

7.2. В. Конфигурация по умолчанию

```
model:
  hidden_size: 64
  num_layers: 2
  num_heads: 2
  max_seq_length: 50
  dropout: 0.5

lowrank:
  rank: 16

training:
  pretrain_epochs: 100
  finetune_epochs: 200
  batch_size: 256
  learning_rate: 0.001

loss_weights:
  aap: 1.0
  mip: 0.2
  map: 1.0
  sp: 0.5
```

7.3. С. Формулы потерь

AAP Loss:

$$\mathcal{L}_{\text{AAP}} = - \sum_{i,a} y_{i,a} \log(\sigma(h_i \cdot W \cdot e_a)) + (1 - y_{i,a}) \log(1 - \sigma(h_i \cdot W \cdot e_a))$$

MIP Loss:

$$\mathcal{L}_{\text{MIP}} = - \sum_i \log(\sigma(h_i \cdot e_{p_i})) - \log(1 - \sigma(h_i \cdot e_{n_i}))$$

MAP Loss:

$$\mathcal{L}_{\text{MAP}} = - \sum_{i,a} y_{m_i,a} \log(\sigma(f(h_i)_a))$$

SP Loss:

$$\mathcal{L}_{\text{SP}} = -\log(\sigma(h^s \cdot h^+)) - \log(1 - \sigma(h^s \cdot h^-))$$

Полная функция потерь:

$$\mathcal{L} = \mathcal{L}_{\text{AAP}} + \alpha \mathcal{L}_{\text{MIP}} + \mathcal{L}_{\text{MAP}} + \beta \mathcal{L}_{\text{SP}}$$

где $\alpha = 0.2$, $\beta = 0.5$ — веса по умолчанию.