

Работа с большими данными



01

Введение. Распределенные вычисления. MapReduce.

02

HDFS. Apache Spark. RDD.

03

Базы данных. Spark SQL. Хранение больших данных.

04

Подробнее о модели вычислений Spark. Знакомство со Scala.

05

Алгоритмы машинного обучения на больших данных. spark.ml.

06

Рекомендательные системы. Виды. Их метрики. Spark.ML

07

Обработка потоковых данных. Structured streaming и интеграция с spark.ml.

08

Модели в продакшен. Управление кластером.

6. Рекомендательные системы. Виды. Их метрики. Spark.ML

План:

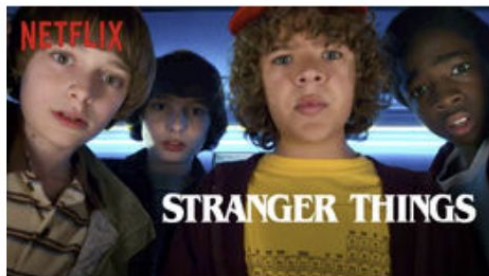
1. Рекомендательные системы. Их задачи.
2. Content based vs Collaborative filtering.
3. Feedback - Implicit vs Explicit.
4. Метрики рекомендательных систем. Как понять, что рекомендации хорошие.
5. Как решать такую задачу. Матричные разложения. ALS.
6. Реализация в Spark.ML.

Рекомендательные системы. Их задачи.

1. Предмет рекомендации – что рекомендуется.
2. Цель рекомендации – зачем рекомендуется.
3. Контекст рекомендации – что пользователь в этот момент делает.
4. Источник рекомендации – кто рекомендует:
5. Степень персонализации.
6. Прозрачность.
7. Формат рекомендации.
8. Алгоритмы.

[Статья вводная про рекомендательные системы](#)

Как выбрать постер?



Виды рекомендательных систем

Можно выделить два основных типа рекомендательных систем. Их, конечно же больше, но мы сегодня будем рассматривать именно эти и в особенности коллаборативную фильтрацию.

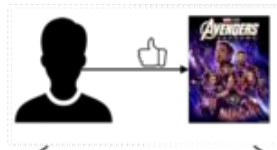
1. Content-based

- Пользователю рекомендуются объекты, похожие на те, которые этот пользователь уже употребил.
- Похожести оцениваются по признакам содержимого объектов.
- Сильная зависимость от предметной области, полезность рекомендаций ограничена.

2. Коллаборативная фильтрация (Collaborative Filtering)

- Для рекомендации используется история оценок как самого пользователя, так и других пользователей.
- Более универсальный подход, часто дает лучший результат.
- Есть свои проблемы (например, холодный старт).

Recommend a movie to this user



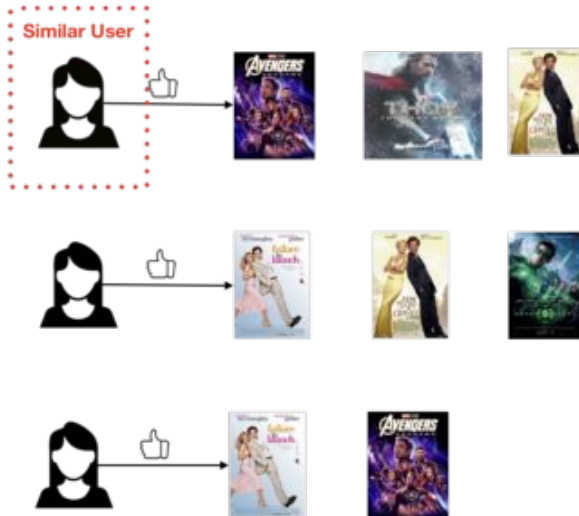
Find similar
users

Find similar
movies

Recommend movies watched by similar user using

Recommend movies similar to Avengers using

Collaborative Filtering







Content Based Filtering



Классическая постановка

В классической постановке задачи всё, что у нас есть — это матрица оценок user-item. Она очень разрежена и наша задача — заполнить пропущенные значения. Обычно в качестве метрики используют RMSE предсказанного рейтинга, но есть мнение, что это не совсем правильно и следует учитывать характеристики рекомендации как целого, а не точность предсказания конкретного числа.

| | M1 | M2 | M3 | M4 | M5 |
|--|----|----|----|----|----|
|  | 3 | 1 | 1 | 3 | 1 |
|  | 1 | 2 | 4 | 1 | 3 |
|  | 3 | 1 | 1 | 3 | 1 |
|  | 4 | 3 | 5 | 4 | 4 |

Feedback - Implicit vs Explicit.

Feedback - как пользователь дает нам знать, что его интересует

Explicit - Стандартный подход к коллаборативной фильтрации на основе матричного факторизации рассматривает записи в матрице пользователь-элемент как явные предпочтения, данные пользователем элементу, например, пользователи, дающие оценки фильмам.

Implicit - Во многих реальных случаях использования обычно имеется доступ только к неявной обратной связи (например, просмотры, клики, покупки, лайки, акции и т. Д.).

Метрики рекомендательных систем. Как понять, что
рекомендации хорошие.

Как понять, что рекомендации хорошие.

Online evaluation

Наиболее предпочтительный способ оценки качества системы — прямая проверка на пользователях в контексте бизнес-метрик. Это может быть CTR, время, проведенное в системе, или количество покупок. Но эксперименты на пользователях дороги, а выкатывать плохой алгоритм даже на малую группу пользователей не хочется, поэтому до онлайн-проверки пользуются оффлайн метриками качества.

Offline evaluation

В качестве метрик качества обычно используют метрики ранжирования, например, MAP@k и nDCG@k.

Offline evaluation





[Отличная статья про метрики качества ранжирования](#)

Еще один вариант, кроме метрик ранжирования - RMSE с ней сегодня в основном и будем работать.




$$RMSE = \sqrt{\frac{1}{|\mathcal{D}|} \sum_{(u,i) \in \mathcal{D}} (\hat{r}_{ui} - r_{ui})^2}$$

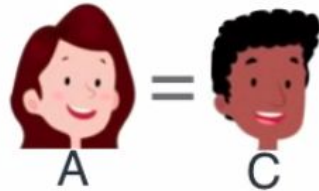
Как решать такую задачу. Матричные разложения. ALS.

Матрица взаимодействий

| | M1 | M2 | M3 | M4 | M5 |
|--|----|----|----|----|----|
|  | 3 | 1 | 1 | 3 | 1 |
|  | 1 | 2 | 4 | 1 | 3 |
|  | 3 | 1 | 1 | 3 | 1 |
|  | 4 | 3 | 5 | 4 | 4 |

Dependent Rows and Columns

| | M1 | M2 | M3 | M4 | M5 |
|---|----|----|----|----|----|
|  | 3 | 1 | 1 | 3 | 1 |
|  | | | | | |
|  | 3 | 1 | 1 | 3 | 1 |
|  | | | | | |



Dependent Rows and Columns

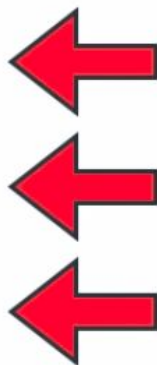
| | M1 | M2 | M3 | M4 | M5 |
|---|----|----|----|----|----|
|  | 3 | | | 3 | |
|  | 1 | | | 1 | |
|  | 3 | | | 3 | |
|  | 4 | | | 4 | |

$$M1 = M4$$







Dependent Rows and Columns

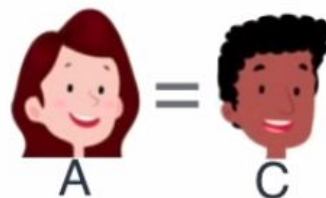
| | M1 | M2 | M3 | M4 | M5 |
|---|----|----|----|----|----|
|  | | | | | |
|  | 1 | 2 | 4 | 1 | 3 |
|  | 3 | 1 | 1 | 3 | 1 |
|  | 4 | 3 | 5 | 4 | 4 |



$$\begin{array}{c} \text{B} \end{array} + \begin{array}{c} \text{C} \end{array} = \begin{array}{c} \text{D} \end{array}$$

Recommender Systems

| | M1 | M2 | M3 | M4 | M5 |
|--|----|----|----|----|----|
|  | 3 | 1 | 1 | 3 | 1 |
|  | 1 | 2 | 4 | 1 | 3 |
|  | 3 | 1 | 1 | 3 | |
|  | 4 | 3 | 5 | 4 | 4 |



Movie 5

Как научиться выделять сразу все зависимости?

Features



Has a Sad Dog



Comedy



Drama



Action



Meryl Streep



Sexy Canadian Ryan





Big Boat







Scary

Matrix Factorization



| |  Comedy |  Action |
|----|---|---|
| M1 | 3 | 1 |
| M2 | 1 | 2 |
| M3 | 1 | 4 |
| M4 | 3 | 1 |
| M5 | 1 | 3 |















| |  Comedy |  Action |
|--|---|--|
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |





=

| | M1 | M2 | M3 | M4 | M5 |
|---|----|----|----|----|----|
|  | 3 | 1 | 1 | 3 | 1 |
|  | 1 | 2 | 4 | 1 | 3 |
|  | 3 | 1 | 1 | 3 | 1 |
|  | 4 | 3 | 5 | 4 | 4 |

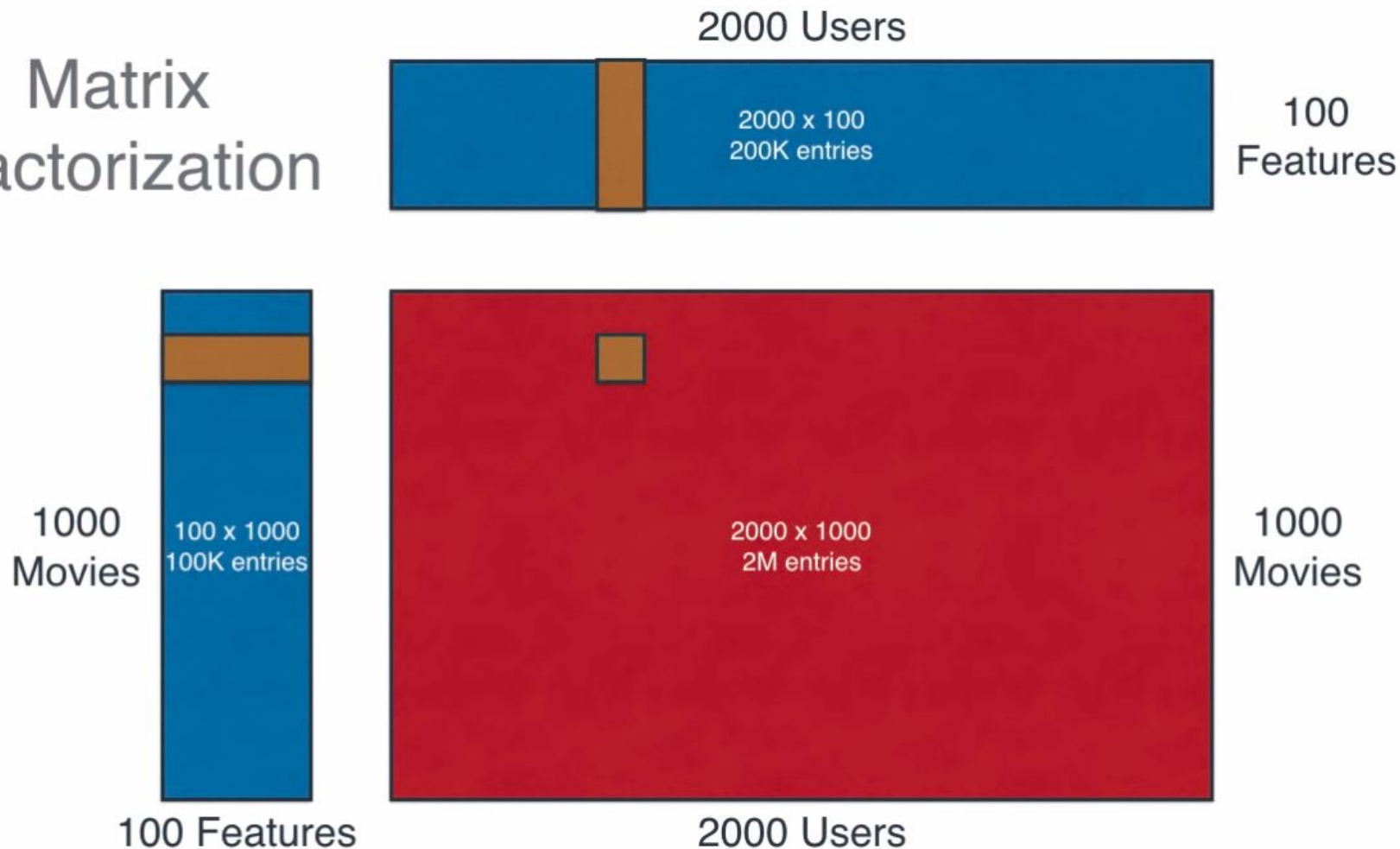
Matrix Factorization

| | M1 | M2 | M3 | M4 | M5 |
|---|----|----|----|----|----|
|  Comedy | 3 | 1 | 1 | 3 | 1 |
|  Action | 1 | 2 | 4 | 1 | 3 |

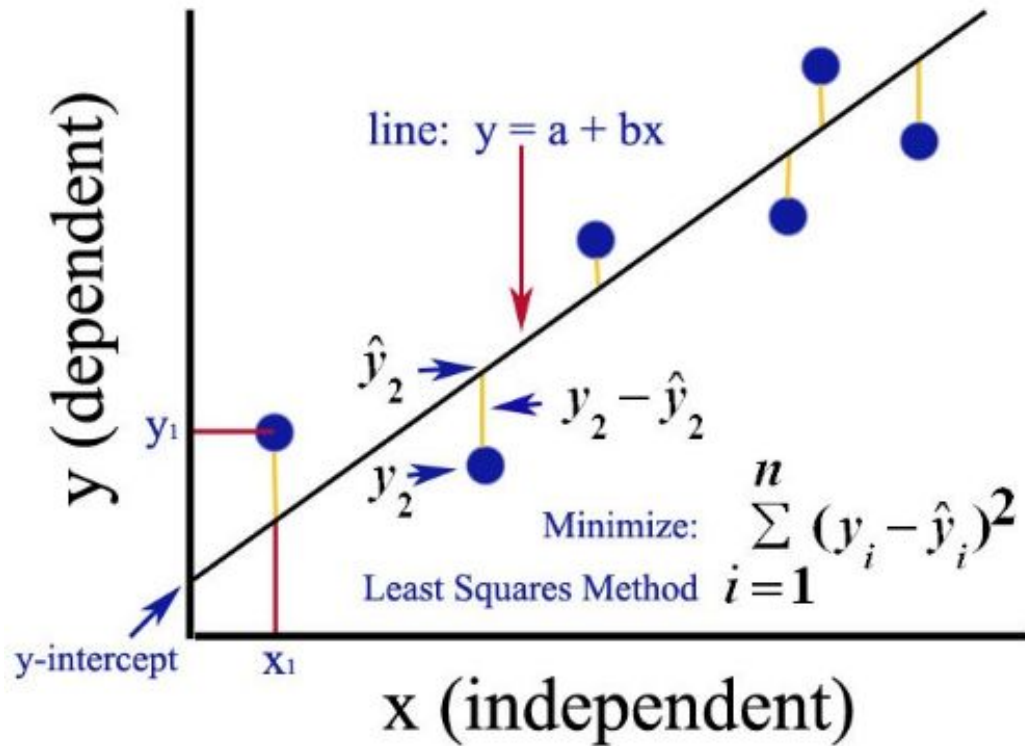
| |  Comedy |  Action |
|---|---|---|
|  A |  |  |
|  B |  |  |
|  C |  |  |
|  D |  |  |

| | M1 | M2 | M3 | M4 | M5 |
|--|----|----|----|----|----|
|  | 3 | 1 | 1 | 3 | 1 |
|  | 1 | 2 | 4 | 1 | 3 |
|  | 3 | 1 | 1 | 3 | 1 |
|  | 4 | 3 | 5 | 4 | 4 |

Matrix Factorization



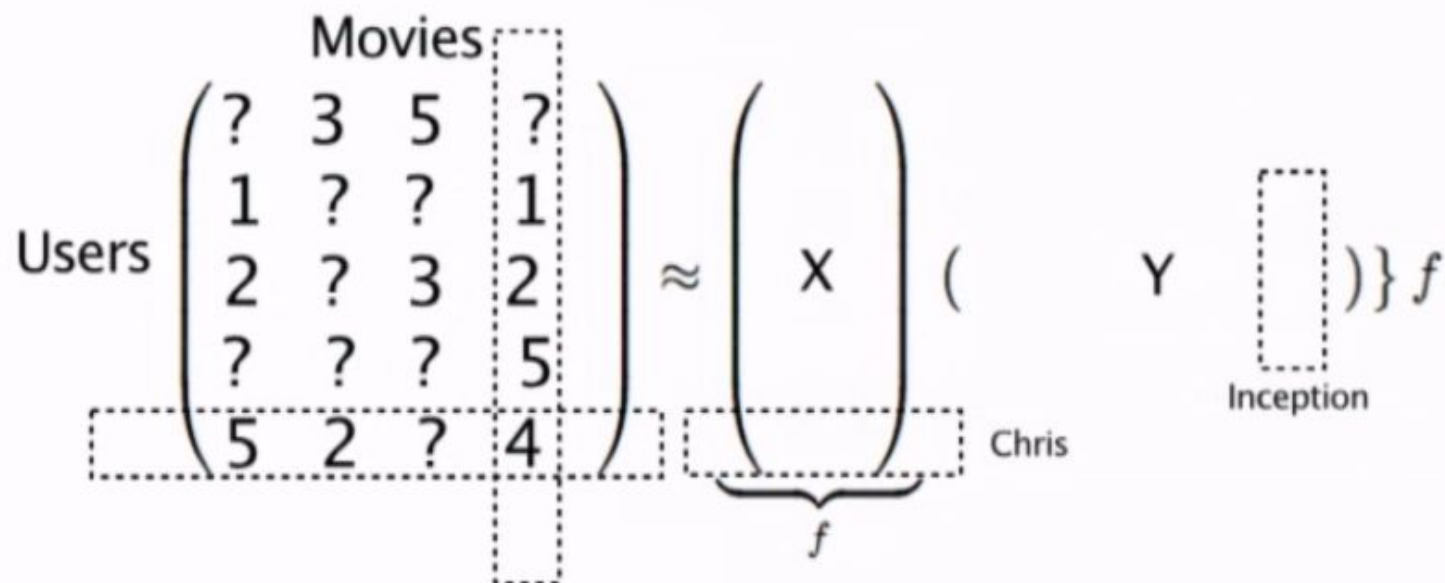
OLS (Ordinary Least Squares)



Метрика, которую оптимизируем

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (p_i - y_i)^2}{n}}$$

ALS (Alternating Least Squares)



$$\min_{x,y} \sum_{u,i} (r_{ui} - x_u^T y_i - \beta_u - \beta_i)^2 + \lambda (\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2)$$

- r_{ui} = user u 's rating for movie i
- x_u = user u 's latent factor vector
- x_i = item i 's latent factor vector
- β_u = bias for user u
- β_i = bias for item i
- λ = regularization parameter

ALS (Alternating Least Squares)

Можно решать последовательность задач минимизации - поочередно фиксируя матрицу пользователей и матрицу фильмов.

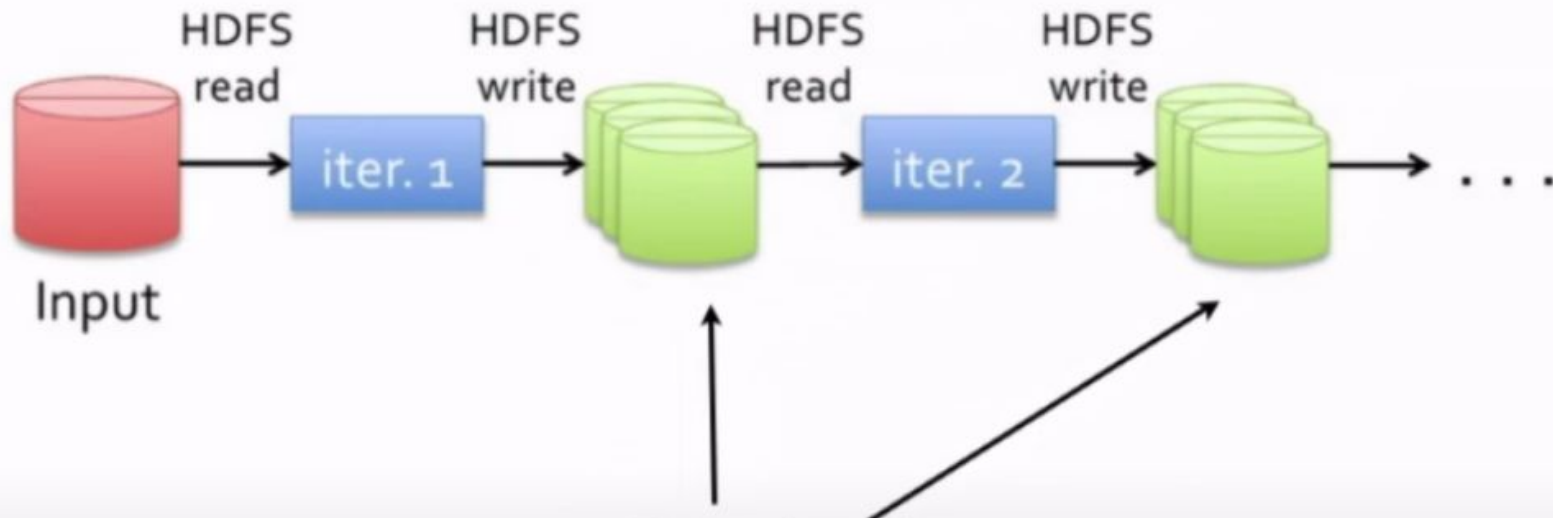
Так задача превращается в последовательность задач OLS (Ordinary Least Squares)

$$\min_{x,y} \sum_{u,i} (r_{ui} - x_u^T y_i - \beta_u - \beta_i)^2 + \lambda (\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2)$$

- r_{ui} = user u 's rating for movie i
- x_u = user u 's latent factor vector
- x_i = item i 's latent factor vector
- β_u = bias for user u
- β_i = bias for item i
- λ = regularization parameter

Реализация в Spark.ML

Hadoop suffers from I/O overhead

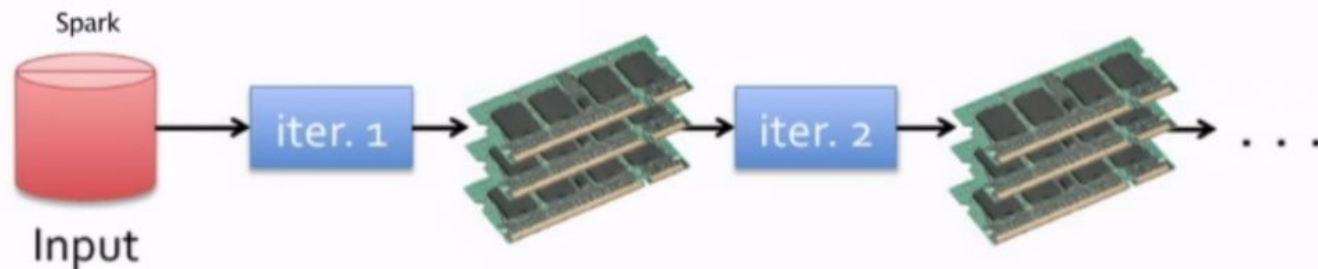


IO bottleneck (input/output performance)

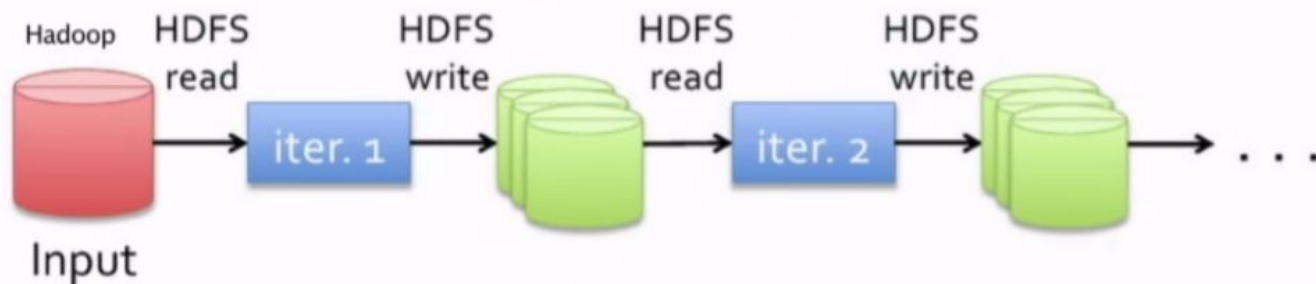
Spark to the rescue!!



26



Vs



Вглубь ALS spark.ml

В видео есть код на scala, а также ребята последовательно рассказывают, как удалось с помощью кэширования в spark-е значительно ускорить все вычисления.

[Видео про рекомендации музыки Spotify с помощью Spark](#)

ALS on Amazon Reviews on 16 Nodes

