

利用PySpark 打造推薦系統

David Chiu
2018/04/25

推薦問題

推薦系統適用於

- 使用者需在眾多的商品選項做取捨:
 - 使用者難以從眾多選項中找他們所想要的，雖然商品搜尋可能可以滿足需求，**但如何提供使用者未知卻需要的商品？**
- 依個人行為與品味做推薦
 - 如果可以找到偏好相似的使用者，便可透過偏好的相似性進行商品推薦

推薦模型

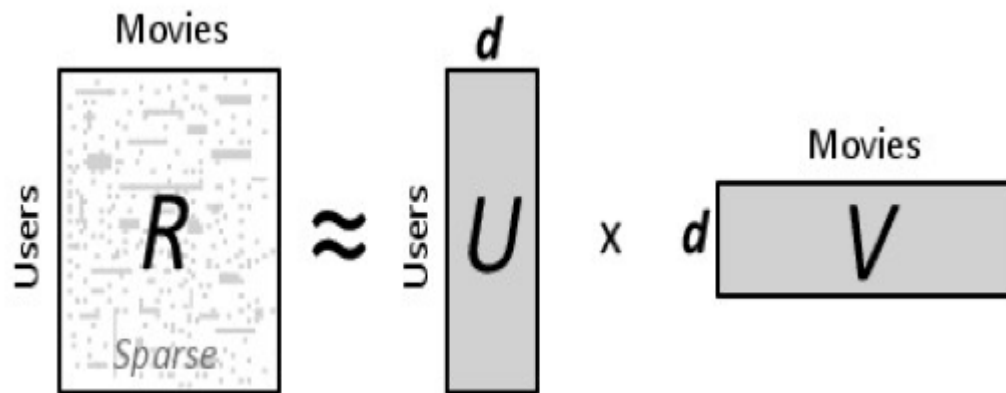
■ 基於相似內容過濾

- 基於商品內容的推薦 (文章標題、文章內容、影片特徵)
- 可以針對使用者檔案所列的喜好做推薦

■ 協同過濾法

- 使用“群眾智慧”找出使用者偏好
- 基於使用者的協同過濾：使用者對類似商品的偏好可推得他們有相同偏好
- 基於商品的協同過濾：商品被類似使用者推薦時，代表性質類似
- 類似最近鄰居法(Nearest Neighbor)

矩陣分解 (Matrix Factorization)



- 假設每位使用者和每樣物品都有自己的特性
- 隱性矩陣分解與顯性矩陣分解

顯性矩陣分解

■ 使用者有對商品表示喜好的評價

Tom, Star Wars, 5

Jane, Titanic, 4

Bill, Batman, 3

Jane, Star Wars, 2

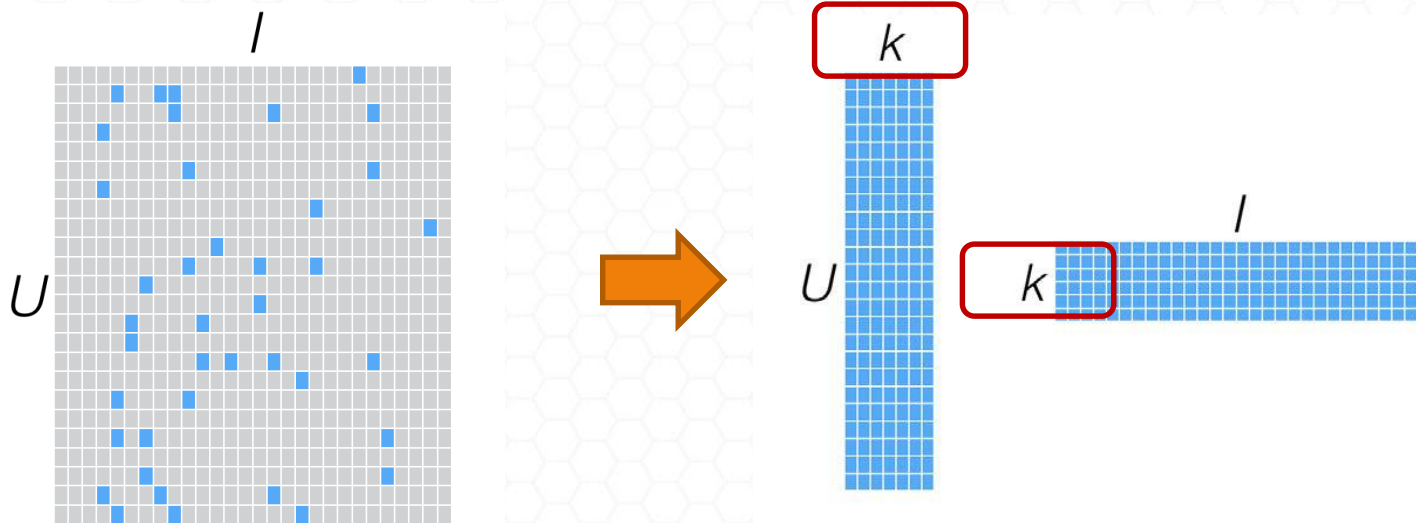
Bill, Titanic, 3



User / Item	Batman	Star Wars	Titanic
Bill	3	3	
Jane		2	4
Tom		5	

低秩矩陣逼近 (Low-rank approximation)

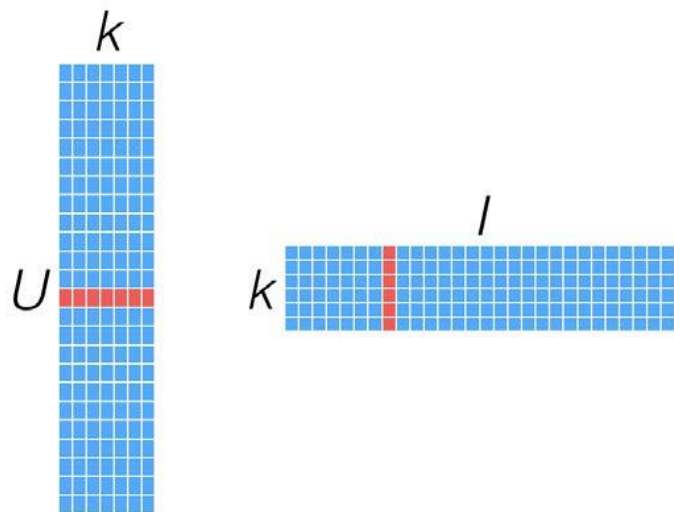
- 使用者對商品評價的稀疏矩陣可分解為使用者對潛在因子(風格、種類)的喜好程度向量與每個商品潛在因子含量(風格、種類)的向量



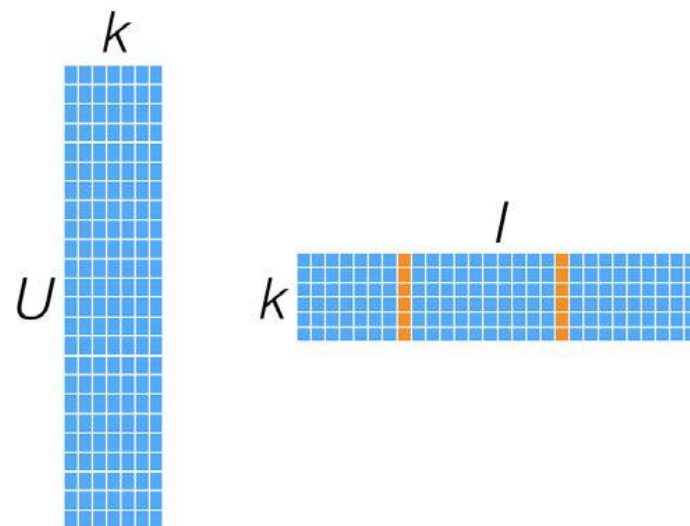
稀疏矩陣 (低秩矩陣)

分解矩陣應用

- 利用使用者與商品向量的乘積計算使用者評分



- 透過商品相似性比對找出相似商品



分解矩陣的優缺點

■ 優點

- 容易計算使用者對商品的評分
- 計算效能高

■ 缺點

- 需要存儲龐大的使用者與商品矩陣
- 訓練過程的計算量高
- 比起最近鄰居法難解讀

隱性矩陣分解

- 使用者並未給商品評分，只能從使用者有沒有買過(或看過)該商品，或以看過次數建立分解矩陣

P

User / Item	Batman	Star Wars	Titanic
Bill	1	1	
Jane		1	1
Tom		1	

有看過則填入 1

C

User / Item	Batman	Star Wars	Titanic
Bill	3	3	
Jane		2	4
Tom		5	

依看過次數填入值

得到使用者對商品的偏好，而非評分

ALS 演算法

■ ALS演算法 (Alternating least squares)

- 解決矩陣分解的最佳化(Optimization)方法
- 可運用在平行運算的框架

■ 演算法

- 隨機產生使用者與商品矩陣
- 固定商品向量以找出最佳使用者向量
 - 根據使用者向量求 RMSE 偏微分為0 的解
- 固定使用者向量以找出最佳商品向量
 - 根據商品向量求 RMSE 偏微分為0 的解
- 迭代迴圈至誤差收斂

Root Mean Squared Error

■ Root Mean Squared Error

- ▣ 可以用來評估矩陣重建的誤差
- ▣ 計算預測值與真實值的差距總和
- ▣ 在ALS 演算法中是用來找尋誤差收斂的模型

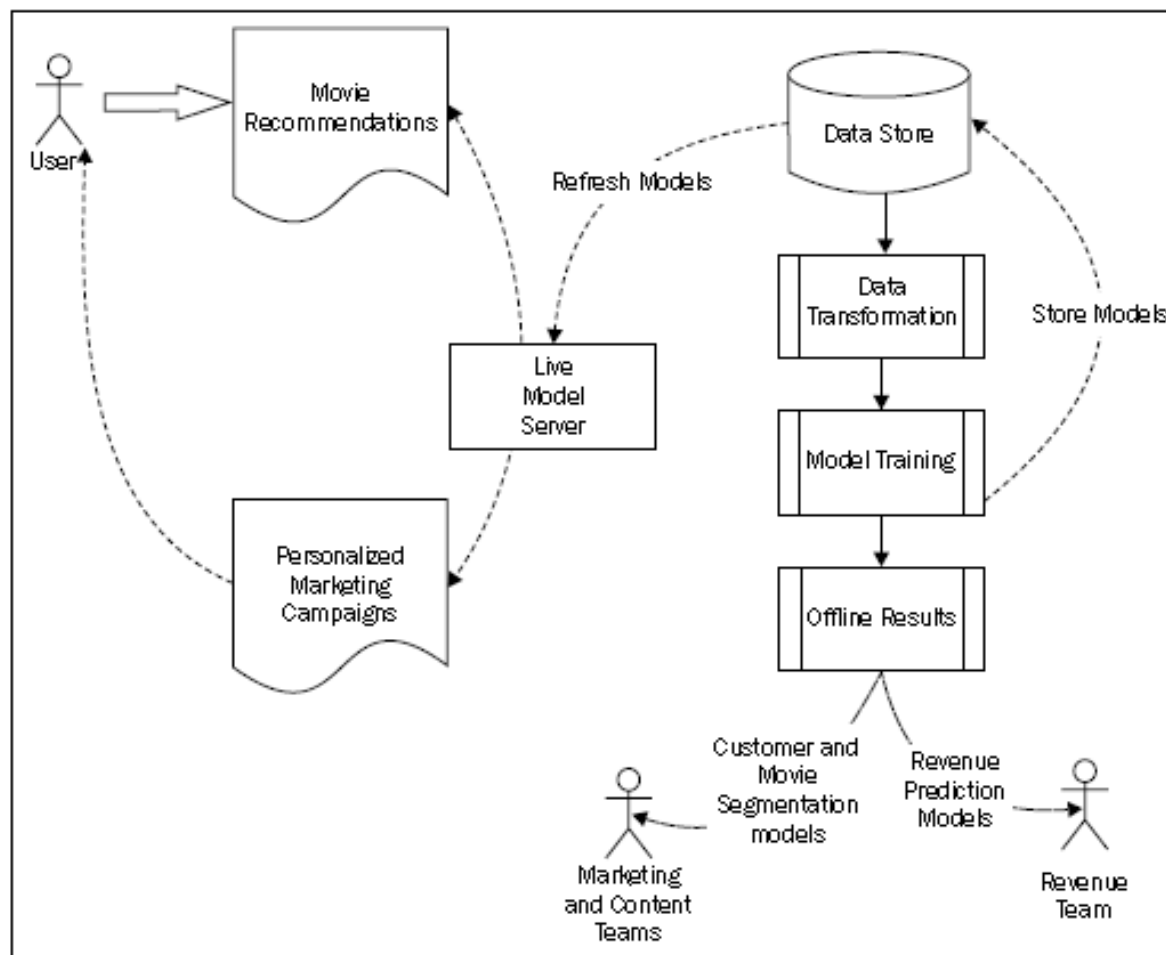
Root Mean Squared Error

$$\text{RMS} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} = \sqrt{\frac{1}{n} \sum_{i=1}^n e_i^2}$$

$$\text{RMS} = \sqrt{\text{MSE}}$$

預測值與真實值的差距

建立推薦模型



MovieStream's future architecture

下載MovieLens 資料

grouplens

[about](#)

[datasets](#)

[publications](#)

[blog](#)

MovieLens

GroupLens Research has collected and made available rating data sets from the MovieLens web site (<http://movielens.org>). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details.

Help our research lab: Please [take a short survey](#) about the MovieLens datasets

MovieLens 100k

Stable benchmark dataset. 100,000 ratings from 1000 users on 1700 movies. Released 4/1998.

- [README.txt](#)
- [ml-100k.zip](#) (size: 5 MB, [checksum](#))
- [Index of unzipped files](#)

MovieLens 1M

Stable benchmark dataset. 1 million ratings from 6000 users on 4000 movies. Released 2/2003.

Datasets

[MovieLens](#)

[HetRec 2011](#)

[WikiLens](#)

[Book-Crossing](#)

[Jester](#)

[EachMovie](#)

讀取MovieLens 資料

```
%pyspark
```

```
rawData = sc.textFile("/tmp/u.data")
```

```
rawData.first() #讀取第一列資料
```

```
#讀取user id, movie id, ratings
```

```
rawRatings = rawData.map(lambda e: e.split())
```

```
rawRatings.take(3)
```

將讀取資料讀進Rating 物件

```
from pyspark.sql import Row

#將資料轉進ratingsRDD 物件
ratingsRDD = rawRatings.map(\
    lambda p: Row(userId=int(p[0]), \
                    movieId=int(p[1]), \
                    rating=float(p[2]), \
                    timestamp=int(p[3])))

# 建立 DataFrame
ratings = spark.createDataFrame(ratingsRDD)

# 將資料區分為訓練與測試資料集
(training, test) = ratings.randomSplit([0.8, 0.2])
```


訓練ALS 模型

```
from pyspark.ml.recommendation import ALS
```

```
als = ALS(rank=50, maxIter=10, regParam=0.01, \  
          userCol="userId", itemCol="movieId", \  
          ratingCol="rating")
```

```
model = als.fit(training)
```

ALS 模型參數

■ Rank

- 於ALS模型中使用多少隱藏的因素 (k)
- 數值越大越好，但使用的記憶體越大
- 設10 ~ 200 是合理的範圍

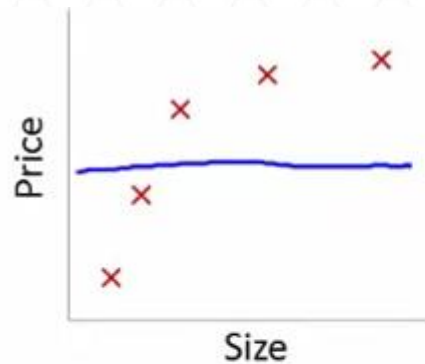
■ Iterations

- 迴圈數，讓重建矩陣跟實際矩陣的誤差收斂

■ regParam

- 懲罰值，須根據Cross-validation 的結果選擇

何謂regParam (Lambda)

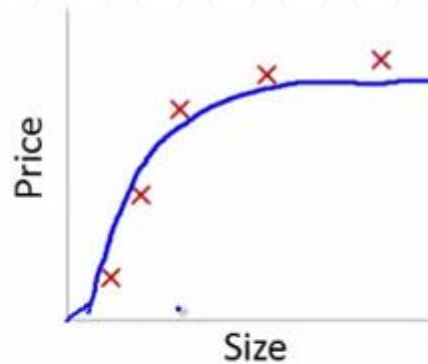


Large λ ←

→ High bias (underfit)

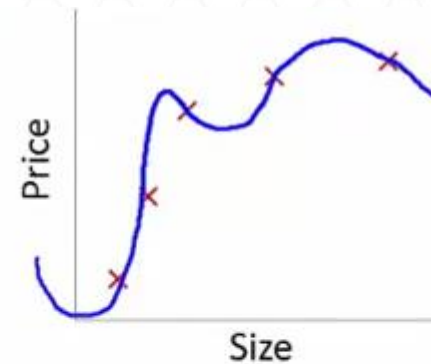
$\lambda = 10000$. $\theta_1 \approx 0, \theta_2 \approx 0, \dots$

$h_{\theta}(x) \approx \theta_0$



Intermediate λ ←

"Just right"



→ Small λ

High variance (overfit)

→ $\lambda = 0$

回傳模型

■ model

- 回傳 MatrixFactorizationModel
- 內含 model.userFactors
- 內含 model.itemFactors

■ 觸動Spark 動作 (Lazy Transformation)

- `model.userFactors.count()`
- `model.itemFactors.count()`

隱性回饋資料

■ alpha

- ▣ 控制信心權重 (confidence weighting) 的基準值 (baseline)

- ▣ 越高的alpha 代表使用者對缺失值(missing data)的資料是代表使用者對該產品沒有偏好

■ 範例

```
model = ALS.trainImplicit(ratings, rank, numIterations, alpha=0.01)
```

求得top-k 的結果

根據使用者推薦

```
userRecs = model.recommendForAllUsers(10)
```

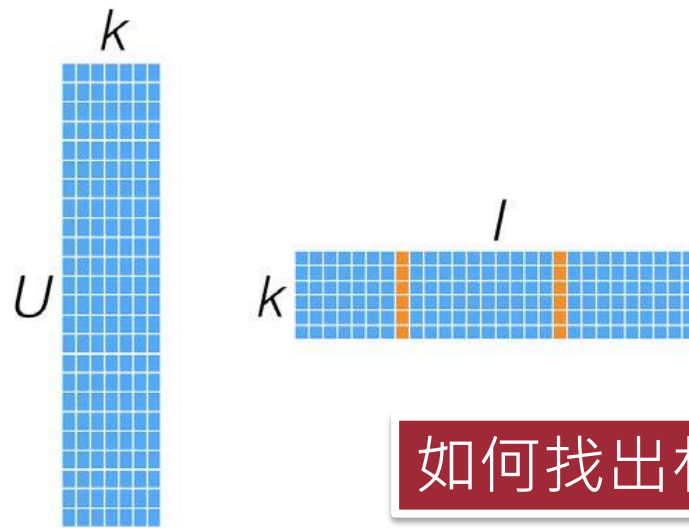
根據電影推薦

```
movieRecs = model.recommendForAllItems(10)
```

透過使用者向量(User Vector)與產品向量(Product Vector)乘積
求得使用者對該商品的評分

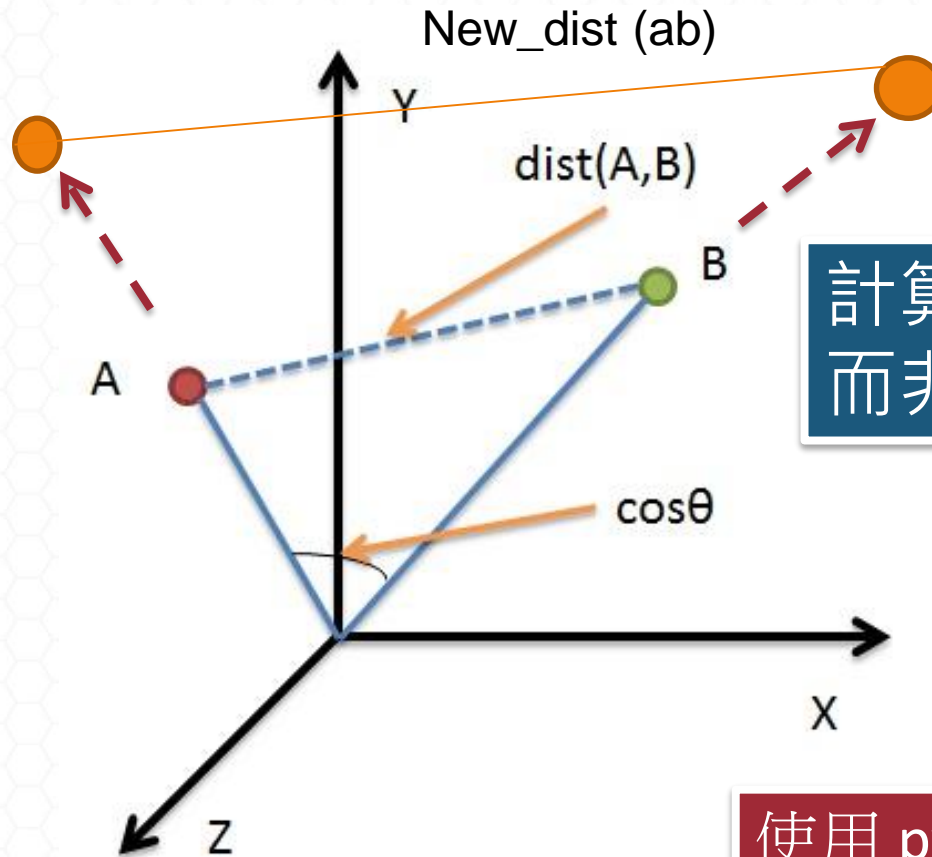
找出相似性的商品

- Pearson Correlation (real value)
- Cosine Similarity (real value)
- Jaccard Similarity (binary vector)



如何找出相似性商品?

計算cosine similarity

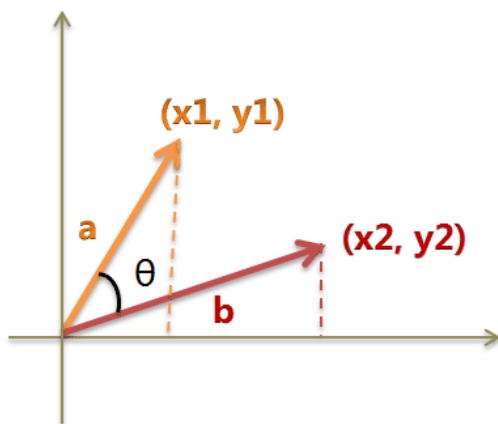


計算相對向量距離
而非絕對距離

使用 `pyspark.ml.linalg`

建立計算cosine similarity 的公式

```
def cosineSimilarity(x, y):  
    return x.dot(y)/(x.norm(2)*y.norm(2))
```



$$\cos\theta = \frac{x_1x_2 + y_1y_2}{\sqrt{x_1^2 + y_1^2} \times \sqrt{x_2^2 + y_2^2}}$$

計算物品間的相似度

```
from pyspark.mllib.linalg import Vectors
features = model.itemFactors.filter('id == 567').select('features')
features_vec = features.rdd.map(lambda e: e['features'][0]).collect()
itemVector = Vectors.dense(features_vec)
cosineSimilarity(itemVector, itemVector)
```

自己跟自己的相似度(Cosine Similarity)為1

計算物品與其他物品的相似度

```
from pyspark.mllib.linalg.distributed import RowMatrix,  
IndexedRowMatrix, IndexedRow  
from pyspark.mllib.linalg.distributed import CoordinateMatrix,  
MatrixEntry  
  
mat = IndexedRowMatrix(model.itemFactors.rdd.map(lambda x:  
IndexedRow(x[0], x[1]))).toBlockMatrix().transpose().toIndexedRowMatrix()  
  
exact = mat.columnSimilarities()  
res = exact.entries.collect()  
for ele in res:  
    print(ele.i, ele.j, ele.value)
```

評估推薦模型的好壞

■ Mean Squared Error

- 計算預測值與實際值的誤差

Root Mean Squared Error

$$\text{RMS} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} = \sqrt{\frac{1}{n} \sum_{i=1}^n e_i^2}$$

$$\text{RMS} = \sqrt{\text{MSE}}$$

■ Mean average precision at K

- 計算top-k 物件的相關性

計算Root Mean Square Error

■ Root Mean Square Error

- ▣ 與真實資料同個單位
- ▣ MSE 開根號

使用內建的RegressionMetrics計算

```
from pyspark.ml.evaluation import RegressionEvaluator

predictions = model.transform(test)
evaluator = RegressionEvaluator(metricName="rmse",
                                labelCol="rating",
                                predictionCol="prediction")
rmse = evaluator.evaluate(predictions)
print("Root-mean-square error = " + str(rmse))
```

The background features a light blue hexagonal grid pattern. Overlaid on this is a large, faint, circular graphic composed of concentric rings and radial lines, resembling a stylized sun or a target. The text "THANK YOU" is centered in a bold, dark blue, sans-serif font.

THANK YOU