{desafío}
latam_

Filtros Colaborativos para recomendaciones con PySpark

Ignacio Soto Zamorano

Big Data Week - SCL 2019

Sobre la charla

Objetivos

- Entender el uso de los filtros colaborativos para generar recomendaciones.
- Identificar los rudimentos del método Alternating Least Squares (ALS).
- Implementar ALS utilizando `pyspark.ml.recommendation.ALS`.
- Generar recomendaciones a partir de un modelo entrenado.
- **Los ejemplos se ejecutarán en modo local.**

Datos

- Yelp Challenge Dataset: https://www.yelp.com/dataset/download

Replicación

- Los archivos se encuentran disponibles en https://github.com/ignaciosotoz/collaborative-filtering-using-pyspark

1. Motivación

Objetivos a responder - Soluciones Alternativas

Recomendación 101

- **Objetivo**: En base a un registro de usuarios e items evaluados, generar sugerencias.

- **Principio basal**: *Si existen dependencias significativas entre usuarios y actividad centrada en el item, podemos utilizarlas para generar nuevas entradas* (Aggarwal 2016, 2).

- La recomendación puede tener objetivos centrados en la **predicción** o en el **ranking** de items.

- Variantes de recomendación:
  - Content-based
  - Knowledge-based
  - Collaborative filtering

Collaborative Filtering

- El filtro colaborativo se puede entender como una generalización al problema de clasificación seminal en Machine Learning.

- Variantes:
  1. **Basados en regresión y decisión:** Árbol de Decisión, Regresión Logística.
  2. **Basados en reglas de asociación:** Algoritmo Apriori.
  3. **Basados en coocurrencias:** Similitudes de Coseno/Jaccard, Algoritmos Aglomerativos, Bayes Ingenuo.
  4. **Basados en modelos de caja negra:** Random Forest, SVM, (Wide and) Deep Learning.
  5. **Basados en factores latentes.**

Modelos de factores latentes

- Los modelos de factores latentes se consideran *state of the art* (Aggarwal 2016).

- Los modelos de factores latentes presentan dos ventajas:
  - Superan el problema de matrices dispersas.
  - Reexpresan la dimensionalidad excesiva en nuevos vectores.

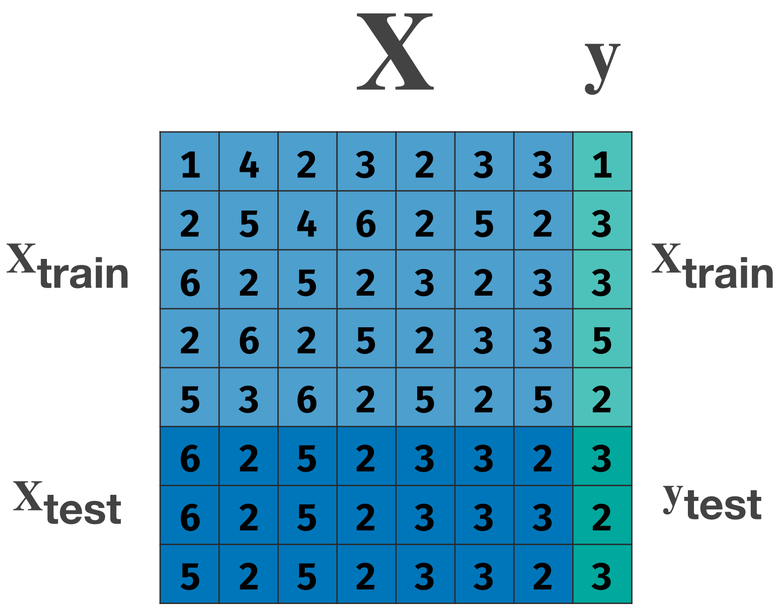
11. Rudimentos

Reducción de Dimensiones - Alternating Least Squares

Problema fundamental

Clasificación

Demarcación clara de los elementos



Filtro colaborativo

No existe demarcación clara
Problema de matriz dispersa

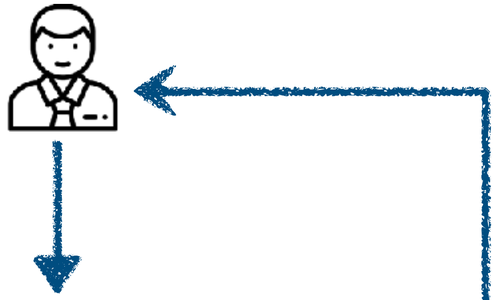
	i_1	i_2	i_3	…	…	…	i_{k-1}	i_k
u_1	1	?	2	?	?	3	3	?
u_2	?	5	?	?	2	?	?	3
u_3	?	?	4	?	?	?	1	?
⋮	2	?	?	?	1	?	?	?
⋮	?	3	?	?	?	?	?	1
⋮	?	?	?	1	?	3	?	?
u_{j-1}	?	3	?	?	5	?	?	?
u_j	5	?	?	4	?	?	2	?

Reducción de Dimensiones

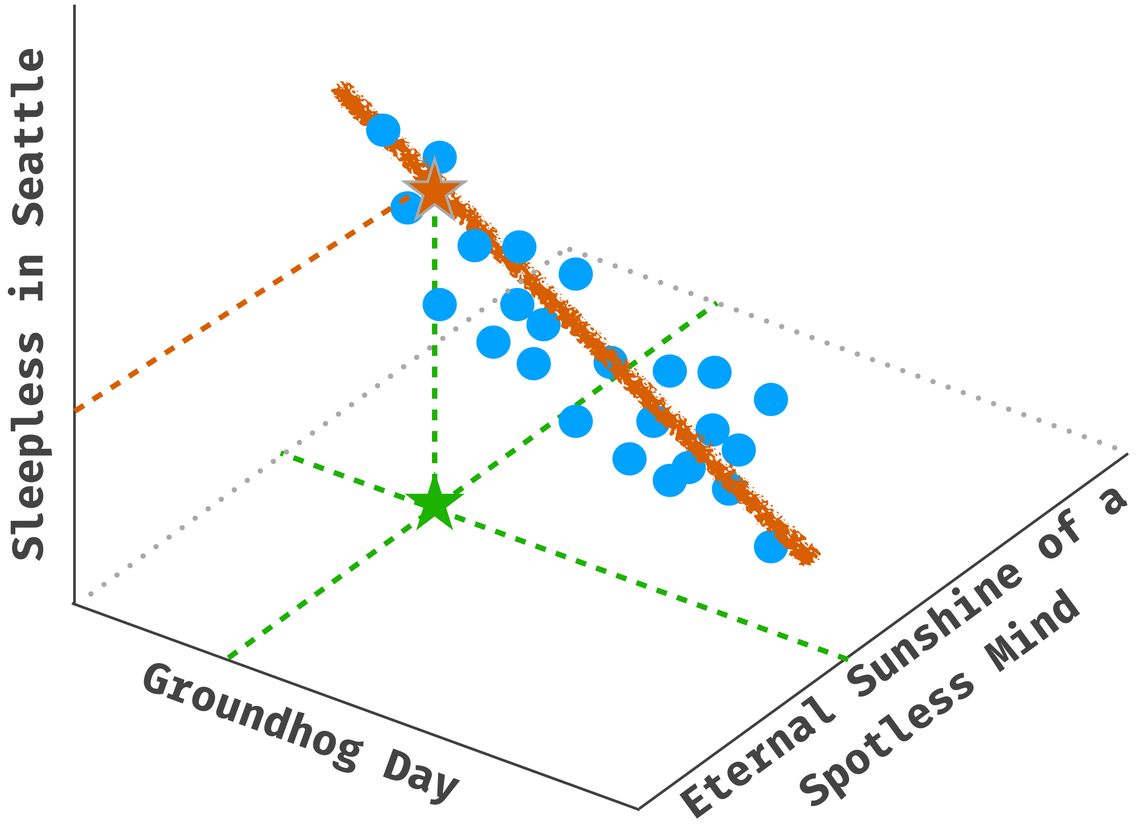
- **Objetivo**: En base a una matriz, generar una representación tratable de los datos mediante un vector latente que represente la relación entre datos.

- Variantes
  - **Principal Components Analysis.**
  - (Truncated) Singular Value Decomposition.
  - Non Negative Matrix Factorization.

Interpretación geométrica



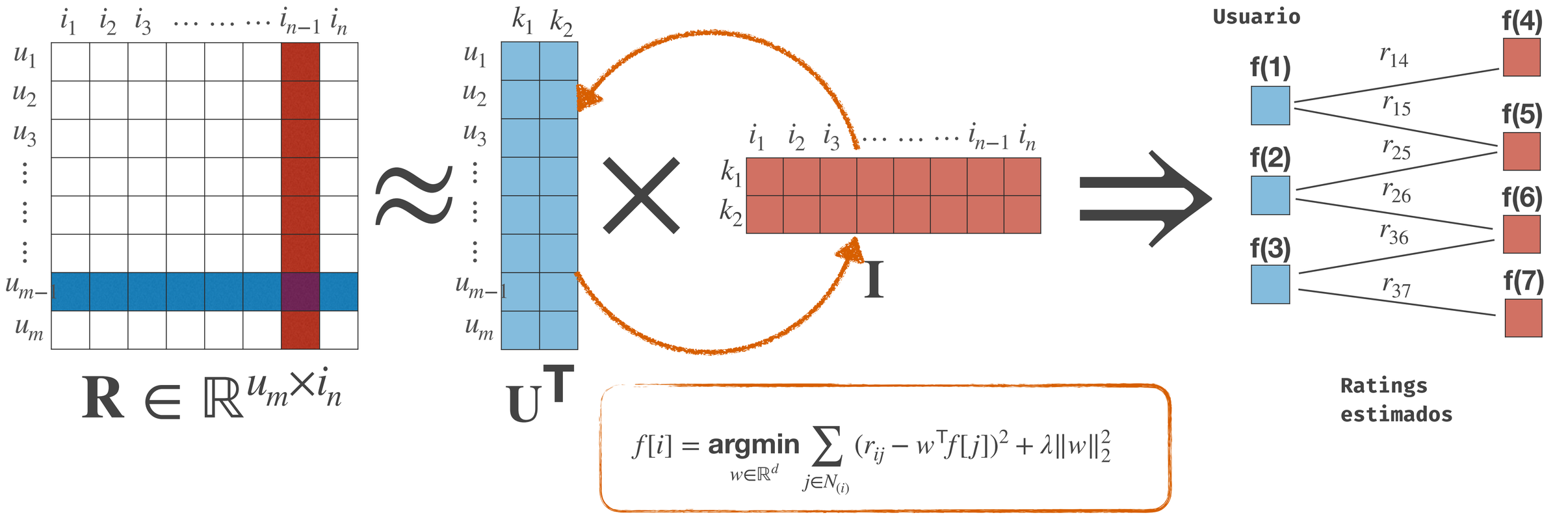
Película	Rating
Groundhog Day	2
Eternal Sunshine of a Spotless Mind	2.5
Sleepless in Seattle	2



Alternating Least Squares

- **Objetivo**: Dado una matriz de ratings $\mathbf{R} \approx \mathbf{U}^{\mathbf{T}}\mathbf{I}$, completar las celdas vacías.
- ALS obtiene un número pequeño de factores $(k \approx 10)$ con los cuales vamos a resumir la matriz de rating a lo largo de usuarios e items.
- En base a estos factores, podemos aproximarnos a la imputación de los ratings a completar.

Intuición de ALS

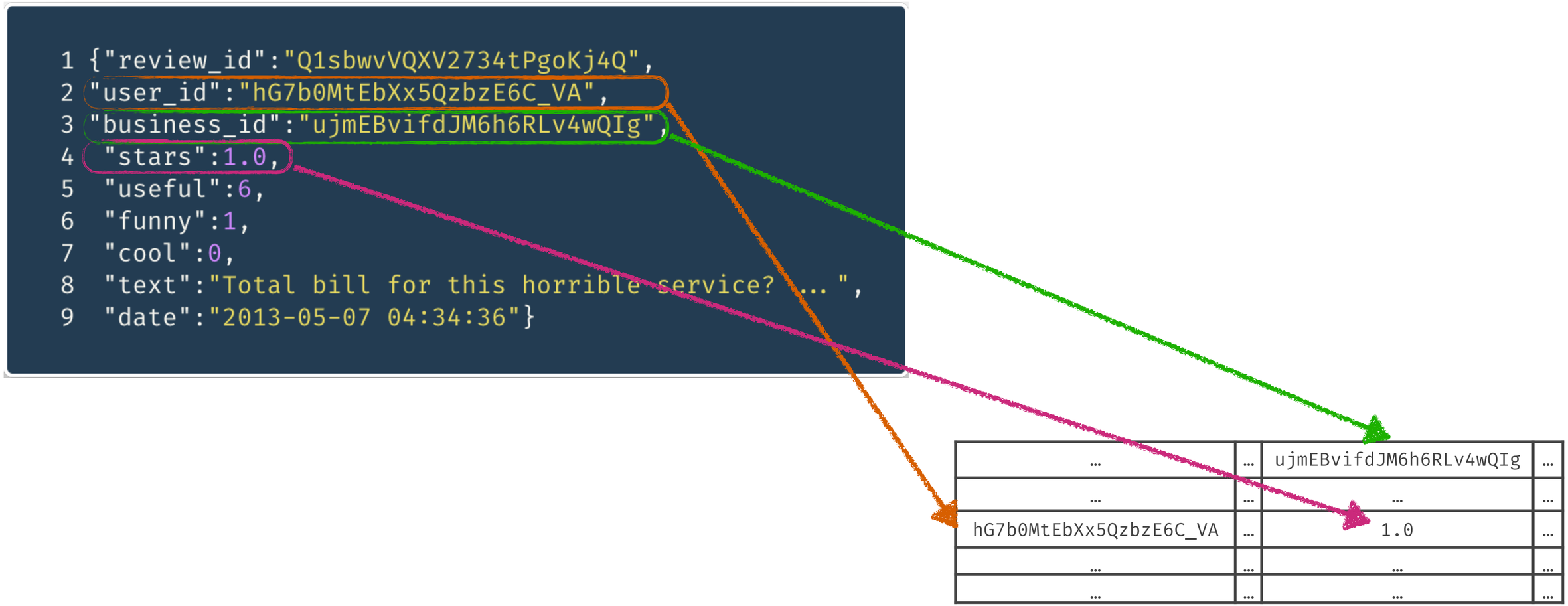


- **Dado una matriz a completar:**
  - Resolvemos un problema de mínimos cuadrados para la matriz **U**, manteniendo **I** fijo.
  - Resolvemos un problema de mínimos cuadrados para la matriz **I**, manteniendo **U** fijo.
  - Iteramos hasta alcanzar convergencia.

III. Práctico

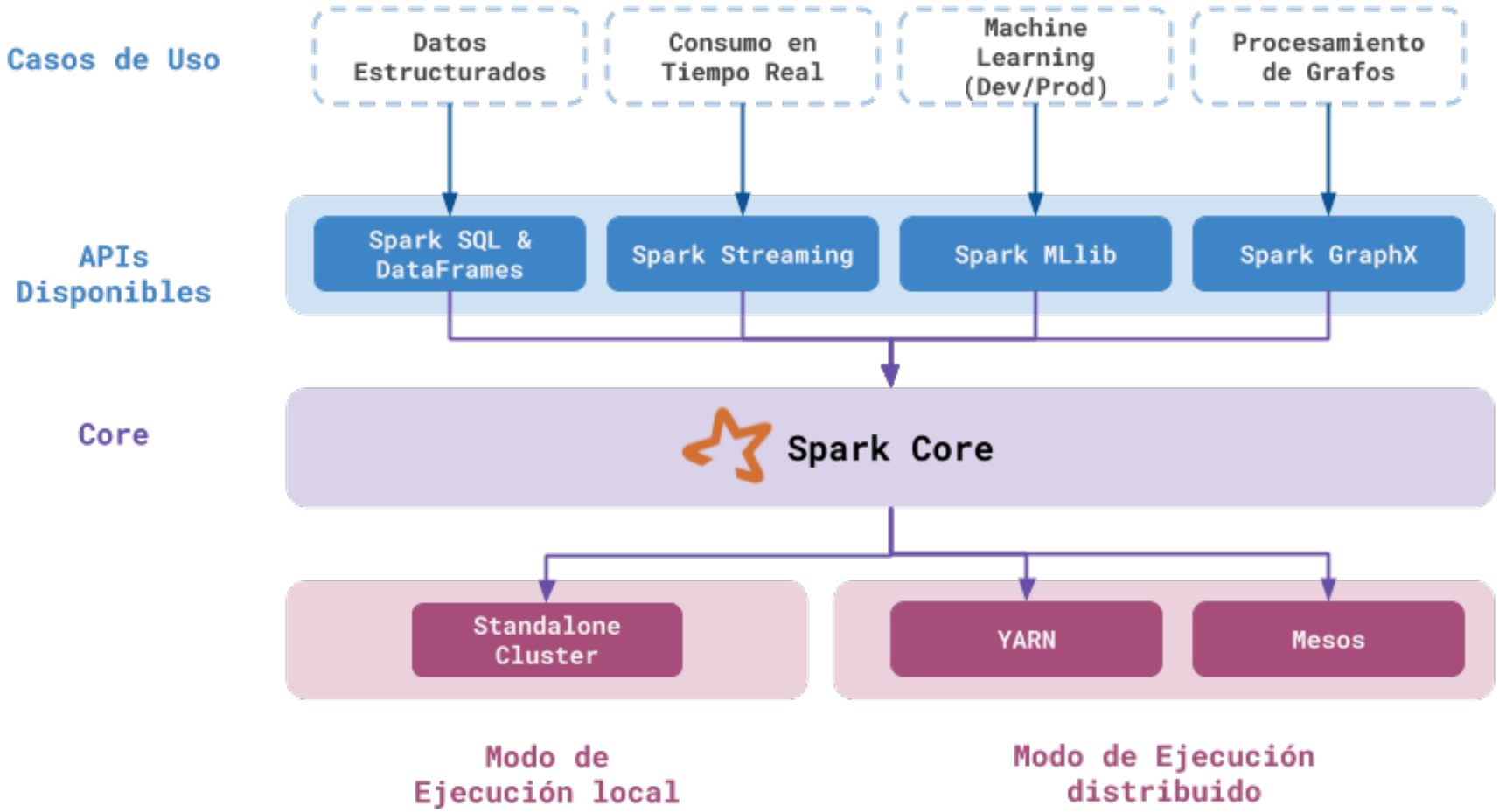
Yelp Data - `pyspark.ml.recommendation.ALS`

Registro ejemplo



Filtros Colaborativos para Recomendaciones con PySpark

Consideraciones



`pyspark.sql.dataframe.DataFrame`

- Son similares a la estructura `pandas.DataFrame` (que se inspira en el `data.frame` de R).
  - Almacenan datos bidimensionales.
  - Columnas con un tipo de dato, filas que representan un registro.

- Tienen el concepto de **lazy evaluation** de los RDD.

- Presenta una buena integración con `pyspark.ml`.

`pyspark.ml`

- Spark es una plataforma de uso general para problemas de Big Data.
- MLlib es un componente estándar que provee de primitivos de Machine Learning sobre Spark.
- `pyspark.mllib`: Versión orientada a RDD.
- `pyspark.ml`: Versión orientada a `pyspark.sql.dataframe.DataFrame`.

SparkSession

```
0 from pyspark.sql import SparkSession
1
2 spark = SparkSession\
3         .builder\
4         .master('local[*]')\
5         .appName('big-data-week')\
6         .enableHiveSupport()\
7         .getOrCreate()
```

Preprocesamiento

```
0 from pyspark.ml.feature import StringIndexer
1 from pyspark.ml import Pipeline
2
3 user_id_indexer = StringIndexer(inputCol="user_id",
4                                 outputCol="user_id_indexed")
5
6 business_id_indexer = StringIndexer(inputCol='business_id',
7                                     outputCol='business_id_indexed')
8
9 pipeline_proc = Pipeline(stages=[user_id_indexer, business_id_indexer])
10
11 yelp_review_proc = pipeline_proc\
12                     .fit(yelp_review)\
13                     .transform(yelp_review)
```

De string a numérico

Concatenación de pasos de trabajo

Definición de la columna a transformar

Concatenación secuencial

Generamos la transformación

Implementamos la transformación

Entrenamiento del modelo

```
0 from pyspark.ml.recommendation import ALS
1
2 train_als = ALS(rank=5,
3                 userCol='user_id_indexed',
4                 itemCol='business_id_indexed',
5                 ratingCol='stars',
6                 regParam=0.01,
7                 coldStartStrategy='drop')
```

Cantidad de vectores latentes a inferir

Identificación de los usuarios

Identificación de los items

Escala de Evaluación

Hiperparámetro de regularización (norma l2)

Estrategia de datos perdidos

```
0 train_als_model = train_als.fit(yelp_review_proc)
```

Entrenamiento del modelo

Aspectos adicionales del modelo

Regularización

- ALS regulariza los factores inferidos para evitar dominancia excesiva en los factores.

- En específico, utiliza norma L2 para disminuir pero no cancelar factores.

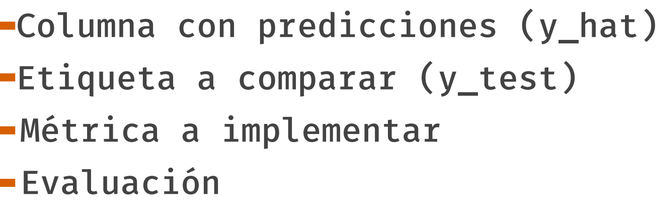
Cold Start Problem

- Los filtros colaborativos funcionan bien en la medida que recopilan información de usuarios existentes.

- Para usuarios nuevos, las predicciones tenderán a ser inexactas.

RegressionEvaluator

```
0 from pyspark.ml.evaluation import RegressionEvaluator
1
2 evaluate_als = RegressionEvaluator(predictionCol='prediction',
3                                    labelCol='stars',
4                                    metricName='rmse')
5 get_rmse = evaluate_als.evaluate(get_predictions)
6 print(f"RMSE promedio: {get_rmse}")
```



RegressionMetrics

```
0 from pyspark.mllib.evaluation import RegressionMetrics
1
2 user_level_prediction = get_predictions\
3                         .select('prediction','stars')\
4                         .rdd\
5                         .map(lambda x: (x[0], x[1]))
6
7 regression_metrics_output = RegressionMetrics(user_level_prediction)
```

RegressionMetrics

```
0 from pyspark.mllib.evaluation import RegressionMetrics
1
2 user_level_prediction = get_predictions\
3                         .select('prediction','stars')\
4                         .rdd\
5                         .map(lambda x: (x[0], x[1]))
6
7 regression_metrics_output = RegressionMetrics(user_level_prediction)
```

Utilizando las predicciones

Seleccionamos lo predicho/observado

Coercionamos a un RDD

Implementamos un map

Ejecutamos

Calibración de Hiperparámetros

4. Agregamos todos los objetos generados en un CrossValidator

```
1 from pyspark.ml.tuning import CrossValidator
2
3 grid_search_als = CrossValidator(estimator=als_instance,
4                                  estimatorParamMaps=params,
5                                  evaluator=eval_rmse,
6                                  numFolds=5)
7 grid_search_cv_train = grid_search_als.fit(data)
```

Instancia del modelo

Instancia de la grilla

Instancia de la métrica

Cantidad de evaluaciones

Referencias

- Aggarwal, Charu. 2016. Recommender Systems. Springer.

- Chambers, Bill; Zaharia, Matei. 2018. Spark, The Definitive Guide: Big Data Processing Made Simple. Sebastopol, CA: O'Reilly Media.

- Karau, Holden; Warren, Rachel. 2017. High Performance Spark: Best Practices for scaling and optimizing Apache Spark. Sebastopol, CA: O'Reilly Media.

- Yang, Xiwang; Guo, Yang; Liu, Yong; Steck, Harald. 2014. A survey of collaborative filtering based social recommender systems. Computer Communications 41: 1-10.

IV. Apéndice

ALS para relleno de matrices

Inicializar U e I

Repetir

$$\forall j = \{1, \cdots, n\} :$$

$$u_j = \left(\sum_{r_{jk} \in r_{j*}} i_k \cdot i_k^{\mathsf{T}} + \lambda \mathbb{I}_f \right)^{-1} \sum_{r_{jk} \in r_{j*}} r_{jk} i_k$$

$$\forall k = \{1, \cdots, m\} :$$

$$i_j = \left(\sum_{r_{jk} \in r_{*k}} u_j \cdot u_j^{\mathsf{T}} + \lambda \mathbb{I}_f \right)^{-1} \sum_{r_{jk} \in r_{j*}} r_{jk} i_j$$

Hasta alcanzar convergencia