

Анализ больших данных с Apache Spark

Лекция 4. Spark MLlib

Мурашкин Вячеслав
2017

<https://github.com/a4tunado/lectures-hse-spark/tree/master/004>

Лекция 4. Spark MLlib

- Spark DataFrame
- Алгоритмы машинного обучения
- Предобработка данных
- MLlib Pipelines
- Подбор оптимальных значений гиперпараметров
- Пример: Кластеризация
- Пример: Пайплайн обработки текста
- Пример: Коллаборативная фильтрация

Spark MLlib

- библиотека распределенного машинного обучения
- реализованы алгоритмы регрессии, классификации, кластеризации и коллаборативной фильтрации
- содержит утилиты предобработки данных и подбора оптимальных значений гиперпараметров алгоритмов (кросс валидация)
- основной API: `pyspark.ml` работает с `DataFrame`
- `pyspark.mllib` - работает с `RDD`, не развивается

Spark DataFrame

- структура данных с типизированными столбцами
- столбцы у DataFrame именованы
- можно создать из `RDD[pyspark.sql.Row]`
- поддерживает SQL запросы
- для работы с DataFrame необходимо создать объект `SparkSession`

Spark DataFrame

```
from pyspark.sql import Row

sc = spark.sparkContext

# Load a text file and convert each line to a Row.
lines = sc.textFile("examples/src/main/resources/people.txt")
parts = lines.map(lambda l: l.split(","))
people = parts.map(lambda p: Row(name=p[0], age=int(p[1])))

# Infer the schema, and register the DataFrame as a table.
schemaPeople = spark.createDataFrame(people)
schemaPeople.createOrReplaceTempView("people")

# SQL can be run over DataFrames that have been registered as a table.
teenagers = spark.sql("SELECT name FROM people WHERE age >= 13 AND age <= 19")

# The results of SQL queries are Dataframe objects.
# rdd returns the content as an :class:`pyspark.RDD` of :class:`Row`.
teenNames = teenagers.rdd.map(lambda p: "Name: " + p.name).collect()
for name in teenNames:
    print(name)

# Name: Justin
```

Spark MLlib - Algorithms

- регрессия / классификация
 - [Linear regression](#)
 - [Logistic regression](#)
 - [Random forest](#)
 - [Gradient-boosted tree](#)
- кластеризация
 - [K-means](#)
 - [Latent Dirichlet allocation \(LDA\) \(topic modelling\)](#)
 - [Gaussian Mixture Model \(GMM\)](#)
- коллаборативная фильтрация (рекомендации)
 - [Alternating least squares \(ALS\)](#)

<http://spark.apache.org/docs/latest/ml-classification-regression.html>

Spark MLlib - Algorithms API

- *Estimator.fit(DataFrame)* - реализует алгоритм обучения, возвращает объект типа *Transformer*
- *Transformer.summary* - информация о результате процесса обучения
- *Transformer.transform(DataFrame)* - возвращает предсказания модели
- *Transformer.save(path)* - сохраняет обученную модель на диск
- *Transformer.load(path)* - загружает сохраненную модель

Предобработка данных и выделение признаков

- Feature Extractors
 - [TF-IDF](#)
 - [Word2Vec](#)
 - [CountVectorizer](#)
- Feature Transformers
 - [Tokenizer](#)
 - [StopWordsRemover](#)
 - [N-Gram](#)
 - [OneHotEncoder](#)
 - [StandardScaler](#)

Пример: Кластеризация. Исходные данные

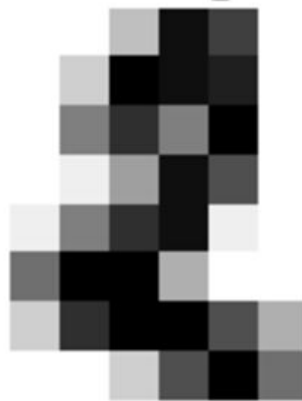
Training: 0



Training: 1



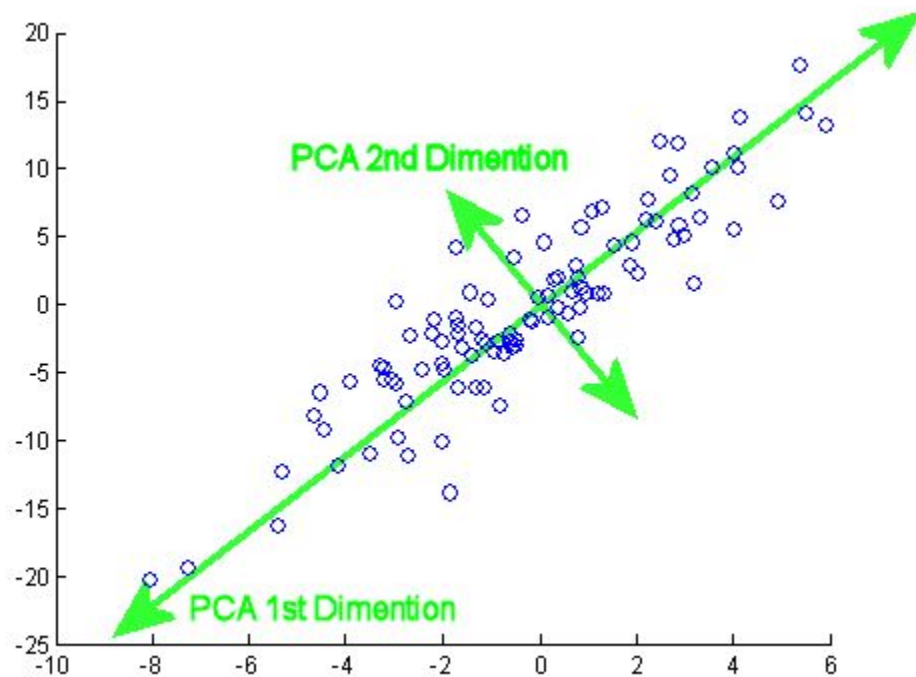
Training: 2




Training: 3














Пример: Кластеризация. PCA



Пример. Кластеризация

 jupyter mlkmeans Last Checkpoint: 14 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help

          Code 

```
In [ ]: # init SparkContext
from pyspark import SparkContext
sc = SparkContext('local', 'K-means')

In [ ]: # load data from csv file
data_rdd = sc.textFile('../data/digits.csv') \
    .map(lambda line: (line.split(',', 1)[0],
                       list(map(float, line.split(',')[1:]))))

In [ ]: # display sample data
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

sample = data_rdd.take(5)
print(sample[0])

fig = plt.figure(figsize=(20, 10))
for i, (y, x) in enumerate(sample, 1):
    subplot = fig.add_subplot(1, len(sample), i)
    plt.imshow(np.array(x).reshape((8, 8)), cmap='gray');
    subplot.set_title('Training: %s' % y);
```

Spark MLlib - Pipelines

- позволяет объединять последовательность алгоритмов в единый интерфейс
- алгоритмы делятся на 2 типа:
 - **Transformer** - как правило создает DataFrame с новым столбцом, в котором доступны результаты преобразования
 - **Estimator** - реализует обучение модели, на выходе объект типа Transformer
- предоставляет единый API для задания параметров алгоритмов

Spark MLlib - Тюнинг моделей

- для подбора оптимальных значений гиперпараметров предоставлено два интерфейса: `CrossValidator` и `TrainValidationSplit`
- эти интерфейсы принимают на вход объекты типа:
 - **Estimator** - алгоритм или пайплайн обучения
 - **Evaluator** - метрика оценки качества алгоритма на отложенных данных
 - `RegressionEvaluator`
 - `BinaryClassificationEvaluator`
 - `MulticlassClassificationEvaluator`
- для задания диапазона поиска значений параметров используется утилита `ParamGridBuilder`

<https://github.com/apache/spark/blob/master/docs/mllib-evaluation-metrics.md>

Пример. Пайплайн обработки текста

Number of reviews: 19411

overall	reviewText
5.0	Good case, solid ...
5.0	This is a terribl...
2.0	I bought this so ...
4.0	works great and c...
4.0	This case is afor...
5.0	This case was by ...
4.0	very good charger...
5.0	Sprint wanted \$30...
5.0	It is Samsung bra...
5.0	good oem car char...

Пример. Пайплайн обработки текста

"It works great. Doesn't heat up like crazy like the other ones..."

Tokenizer

["It", "works", "great", "Doesn't", "heat", "up", "like", "crazy", "like",
"the", "others", "ones"]

StopWordsRemover

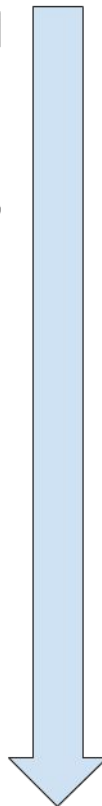
["works", "great", "heat", "up", "like", "crazy", "like", "ones"]

Ngram

[("works", "great"), ("great", "heat"), ("heat", "up"), ("up", "like"),
("like", "crazy"), ("crazy", "like"), ("like", "ones")]

HashingTF

[0, 1, 0, ... 1, 1, 0]



Пример. Пайплайн обработки текста

Term	Index
John	1
likes	2
to	3
watch	4
movies	5
Mary	6
too	7
also	8
football	9

- *John likes to watch movies.*
- *Mary likes movies too.*
- *John also likes football.*

John	likes	to	watch	movies	Mary	too	also	football
1	1	1	1	1	0	0	0	0
0	1	0	0	1	1	1	0	0
1	1	0	0	0	0	0	1	1

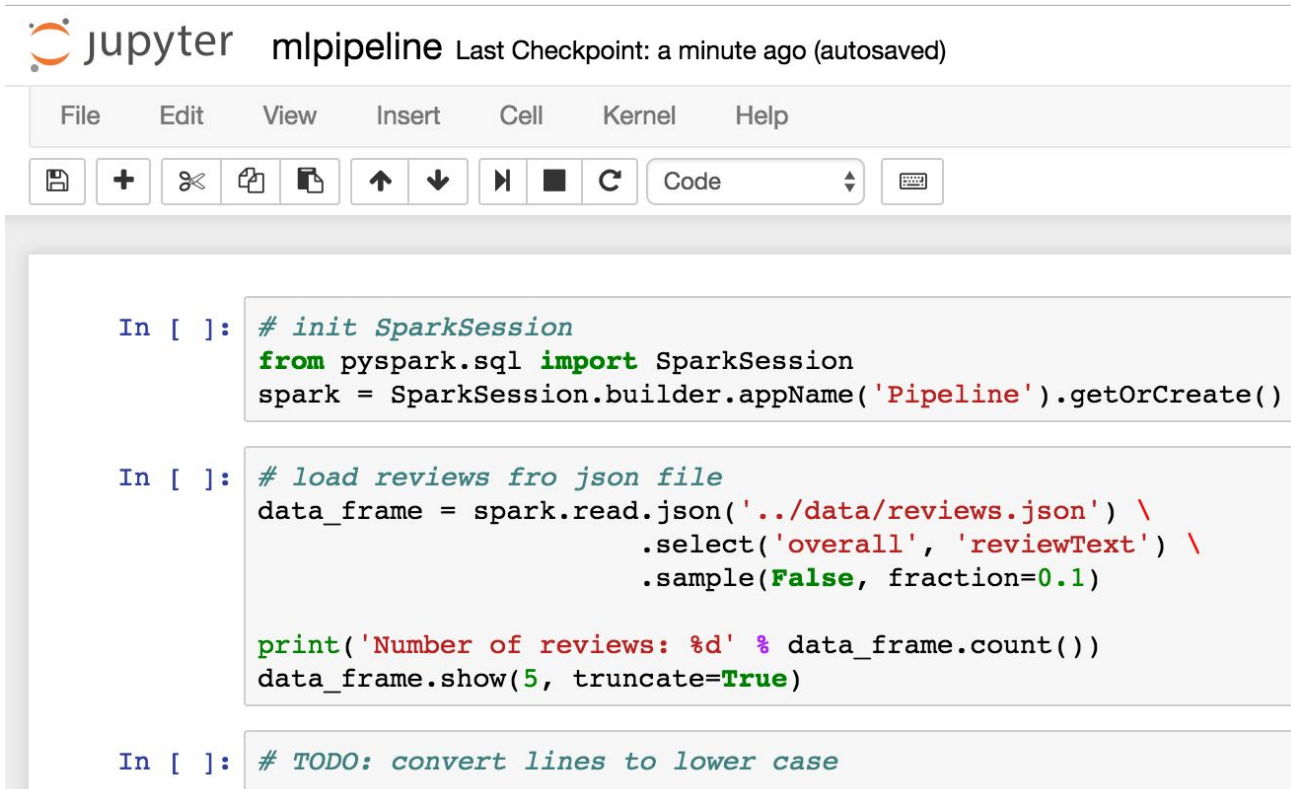
Пример. Пайплайн обработки текста

```
function hashing_vectorizer(features : array of string, N : integer):  
    x := new vector[N]  
    for f in features:  
        h := hash(f)  
        x[h mod N] += 1  
    return x
```

Пример. Пайплайн обработки текста

```
uint32_t jenkins_one_at_a_time_hash(const uint8_t* key, size_t length) {  
    size_t i = 0;  
    uint32_t hash = 0;  
    while (i != length) {  
        hash += key[i++];  
        hash += hash << 10;  
        hash ^= hash >> 6;  
    }  
    hash += hash << 3;  
    hash ^= hash >> 11;  
    hash += hash << 15;  
    return hash;  
}
```

Пример. Пайплайн обработки текста



The screenshot shows a Jupyter Notebook interface with the title "mlpipeline" and a status bar indicating "Last Checkpoint: a minute ago (autosaved)". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". Below the menu bar is a toolbar with icons for saving, adding, deleting, copying, pasting, and navigating between cells. The main area contains three code cells. The first cell initializes a SparkSession. The second cell loads reviews from a JSON file, selects specific columns, and samples the data. The third cell is a TODO comment for converting lines to lower case.

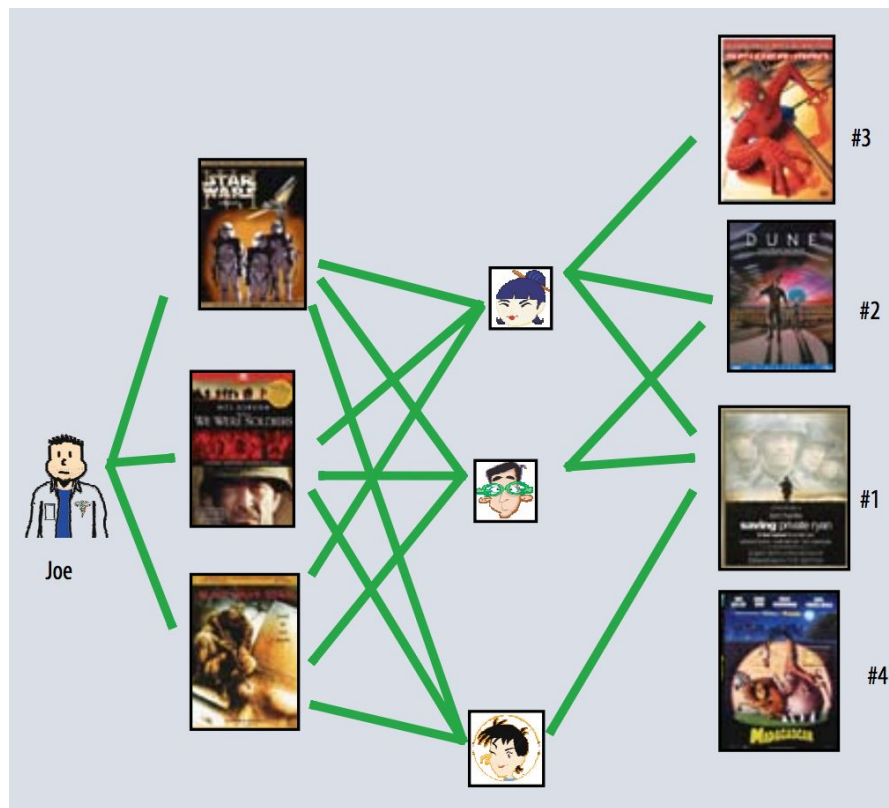
```
In [ ]: # init SparkSession
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('Pipeline').getOrCreate()

In [ ]: # load reviews fro json file
data_frame = spark.read.json('../data/reviews.json') \
    .select('overall', 'reviewText') \
    .sample(False, fraction=0.1)

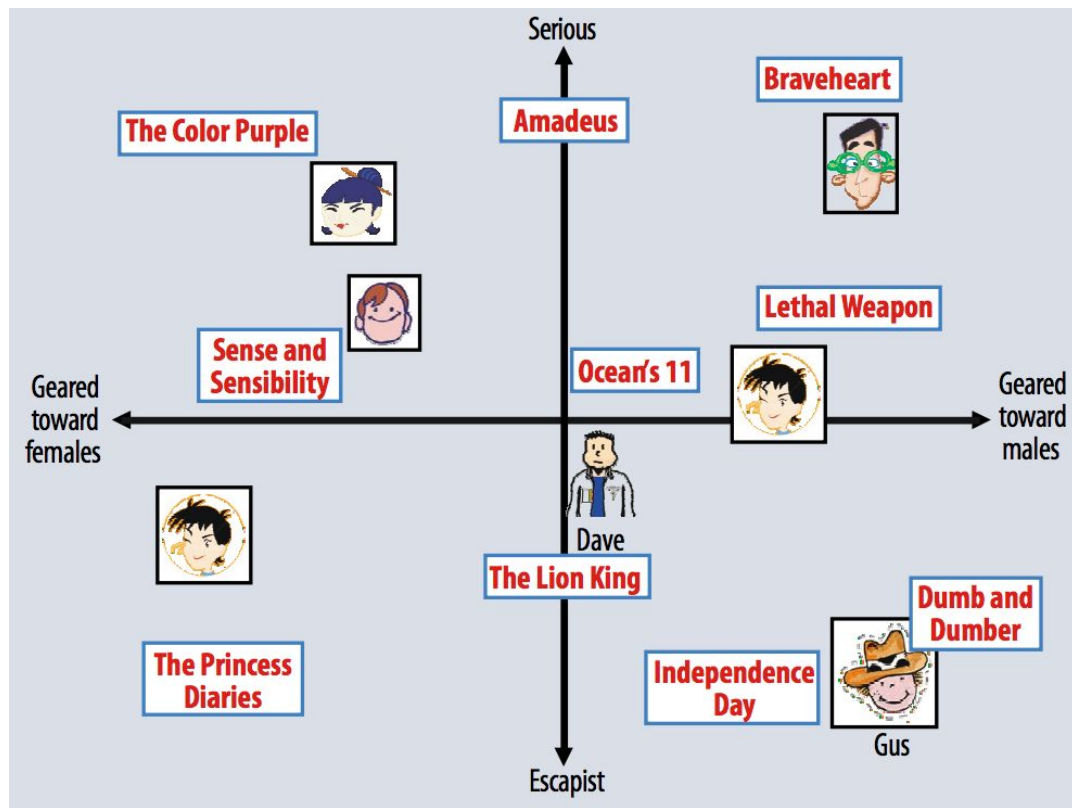
print('Number of reviews: %d' % data_frame.count())
data_frame.show(5, truncate=True)

In [ ]: # TODO: convert lines to lower case
```

Пример. Рекомендации



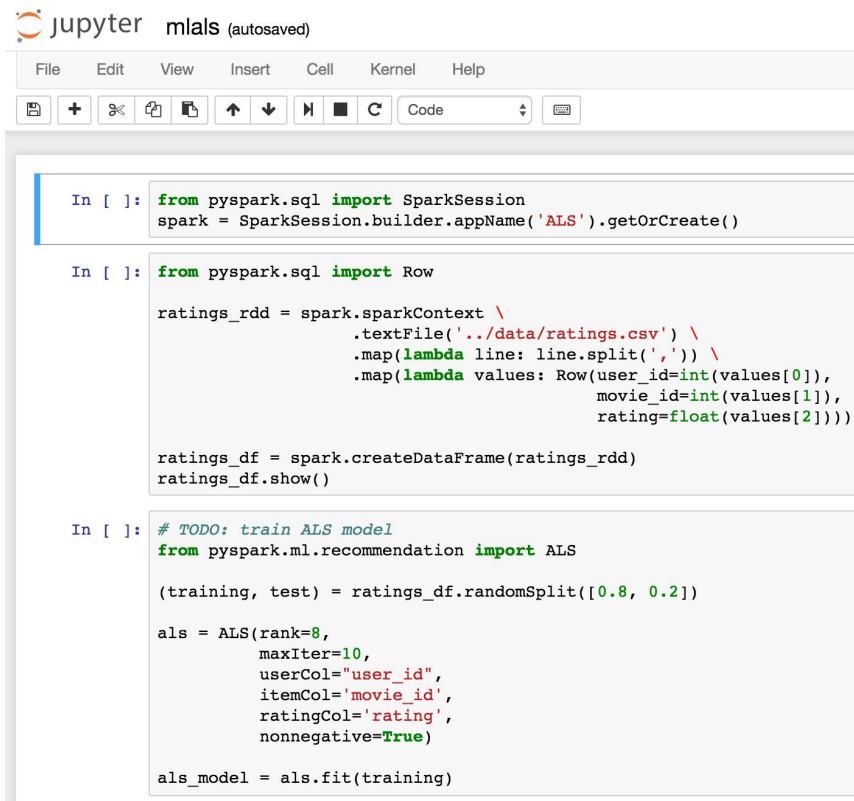
Пример. Рекомендации



Пример. Рекомендации

$$\min_{q^*, p^*} \sum_{(u,i) \in K} (r_{ui} - q_i^T p_u)^2 + \lambda (\|q_i\|^2 + \|p_u\|^2)$$

Пример. Рекомендации



The image shows a Jupyter Notebook interface with the title 'mlals (autosaved)'. The interface includes a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. Below the menu is a toolbar with icons for saving, adding cells, undo, redo, and other standard Jupyter actions. The main area contains three code cells. The first cell imports SparkSession from pyspark.sql and creates a SparkSession named 'ALS'. The second cell imports Row from pyspark.sql, reads a CSV file 'ratings.csv', maps it to Row objects, and shows the resulting DataFrame. The third cell imports ALS from pyspark.ml.recommendation, splits the ratings DataFrame into training and test sets, and fits the ALS model to the training data.

```
In [ ]: from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('ALS').getOrCreate()

In [ ]: from pyspark.sql import Row

ratings_rdd = spark.sparkContext \
    .textFile('../data/ratings.csv') \
    .map(lambda line: line.split(',')) \
    .map(lambda values: Row(user_id=int(values[0]),
                           movie_id=int(values[1]),
                           rating=float(values[2])))

ratings_df = spark.createDataFrame(ratings_rdd)
ratings_df.show()

In [ ]: # TODO: train ALS model
from pyspark.ml.recommendation import ALS

(training, test) = ratings_df.randomSplit([0.8, 0.2])

als = ALS(rank=8,
          maxIter=10,
          userCol="user_id",
          itemCol='movie_id',
          ratingCol='rating',
          nonnegative=True)

als_model = als.fit(training)
```

Полезные материалы

- Machine Learning Library (MLlib) Guide
<http://spark.apache.org/docs/latest/ml-guide.html>
- pyspark.ml package
<http://spark.apache.org/docs/latest/api/python/pyspark.ml.html>
- Evaluation Metrics - RDD-based API
<https://github.com/apache/spark/blob/master/docs/mllib-evaluation-metrics.md>
- Feature hashing
https://en.wikipedia.org/wiki/Feature_hashing
- Matrix factorization techniques for recommender systems
<https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf>