Spark for data science

→ Data is getting big & we need ways to handle it.

Lesson overview

1. Big data ecosystem
2. Data wrangling
3. Debugging & optimisation
4. Spark ML

The power of spark

Big data → lots of tools (ecosystem)

↖ ? what is this

- Data that cannot be processed on a single machine

→ important numbers

CPU .4 us ← "Brain" of pc
Memory 100 us ← "Scratch Paper"
Storage 16 ms ← "File cabinet"
Network 150 ms ← "outside world"

CPU is FAST
problem: loading data takes longer than time taken to process
↳ we'll also lose data when pc is turned off.
↳ is also expensive

Storage is slower still, but it is cheaper

Network even slower
← tends to be bottleneck when working w/ big data

Spark ↓ data shuffling

When should you use spark?

↳ too big to process on 1 pc

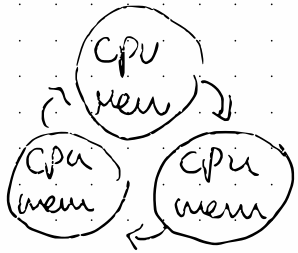
∴ split the data & distribute it

↳ email? ↳ diff versions
↳ slow
↳ not controlled

→ Do it to prevent "thrashing"
↳ loading data in and out of memory

→ even if data is bigger than your local memory, you can still load it using spark to handle the data in batches

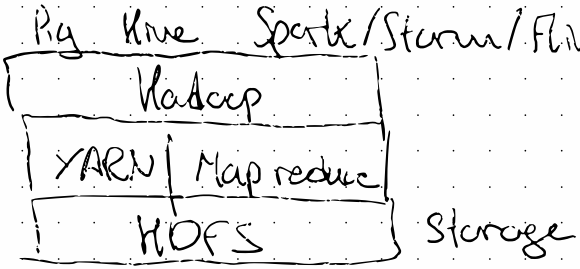
↳ Distributed computing



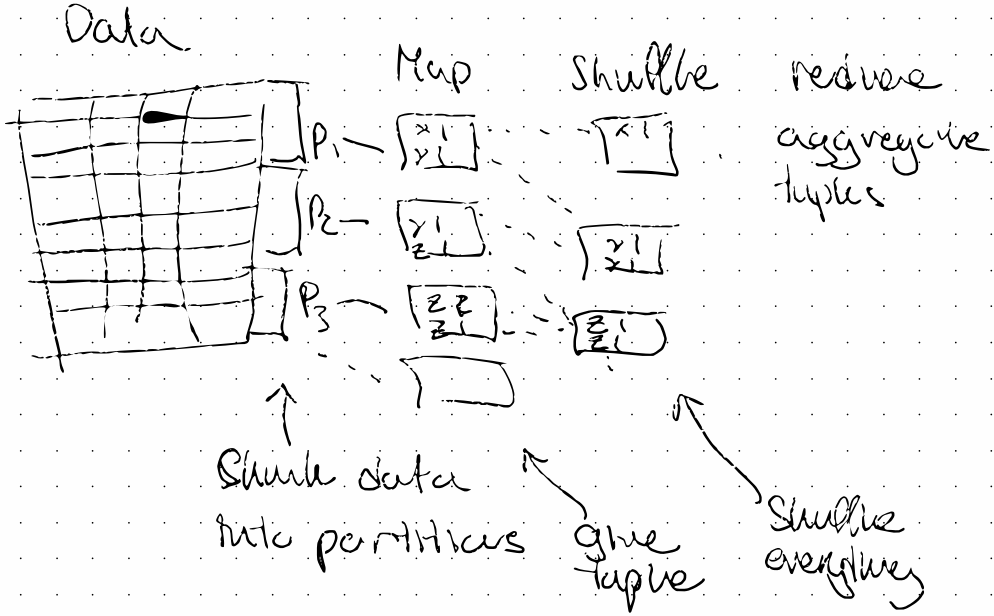
↳ Parallel comp.

Memory ↳ CPU
↳ CPU
↳ CPU

Hadoop framework



Mapreduce



Data wrangling w/ spark

Spark ← functional programming paradigm (scala)

↖ access through Python via API (Pyspark)

Why use functional programming?

- gets name from "functions" in mathematics

Not the same as procedural prog.

ex. If you run a counter on an iterator many times you will get a different output.

- Functions should not access global variables
- No changes made to input data either.
- Chaining of functions together

How to make a spark script

↳ Parallelize dataset
↳ use a lot of map functions (w/ lambda fncs.)
↳ only runs at the last second

basic workflow:

```
import pyspark as ps

sc = ps.SparkContext(name)

rdd = sc.parallelize(data)

rdd.map(func).collect()
```

↑ can be a fnc or lambda

↑ only runs at this point

↳ select from rdd (resilient dist. dataset) using common keywords

rdd.where
rdd.select ? order doesn't matter
:
↳ or use rdd.sql( ) for SQL input

Data formats

↳ CSV, JSON, XML, HTML, ...

Distributed Data Stores

↳ Stores data across a cluster (HDFS)

Adds redundancy so if 1 node dies then no data is lost

Spark Sessions

Spark
↓ Program

Spark Context
↳ Spark Conf in order to configure settings.

Imperative vs Declarative Programming

↳ Python
"How?" ← Path to results

↳ SQL
"What" ← Results

this means that when you tell SQL what you want to see, it'll know exactly how best to make it.

↳ adv. does one thing well

↳ disadv. only 1 thing done well.

Aggregations

done w/ rdd.agg( )

↑ dict w/
{ col : fnction }

rename to useable w/

rdd.withColumnRenamed('agg(col)', <new>)

Machine Learning w/ Spark

Libraries → Spark ML
→ Spark MLLib ← Slowly getting deprecated

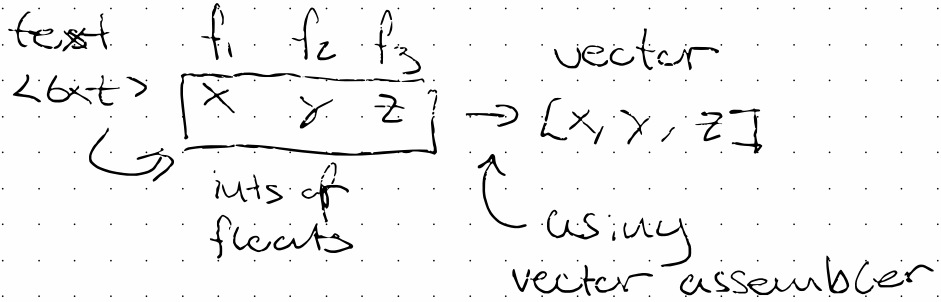
Feature extraction

↳ information needs to be extracted from data to make features

↳ Also needs to be normalised & put into the correct format.

Spark is a little weird...

↳ each row needs to become a vector to be used in functions



General Workflow for ML

↳ Wrangle data

↓

turn to vec & label col

↓

place into ML algorithm

train/test splits?

k-fold CV?

Pipelines

Spark ML has similar pipeline capabilities to sklearn

- transformers ← feature extraction
- regressors ← ML algorithm

↳ pipeline (stages = <list of transformers>)
w/ est at the end

Model selection & tuning

param grid
↳ Has params to be searched

evaluator
↳ evaluates performance
↳ found in pyspark.ml.tuning