Ensembles in Apache Spark

Big Data II

Diego García (djgarcia@decsai.ugr.es) 10/04/2018

Outline

- **Ensembles**
- Apache Spark
- MLlib

Ensembles

Ensembles are methods that combine a set of base classifiers to make predictions

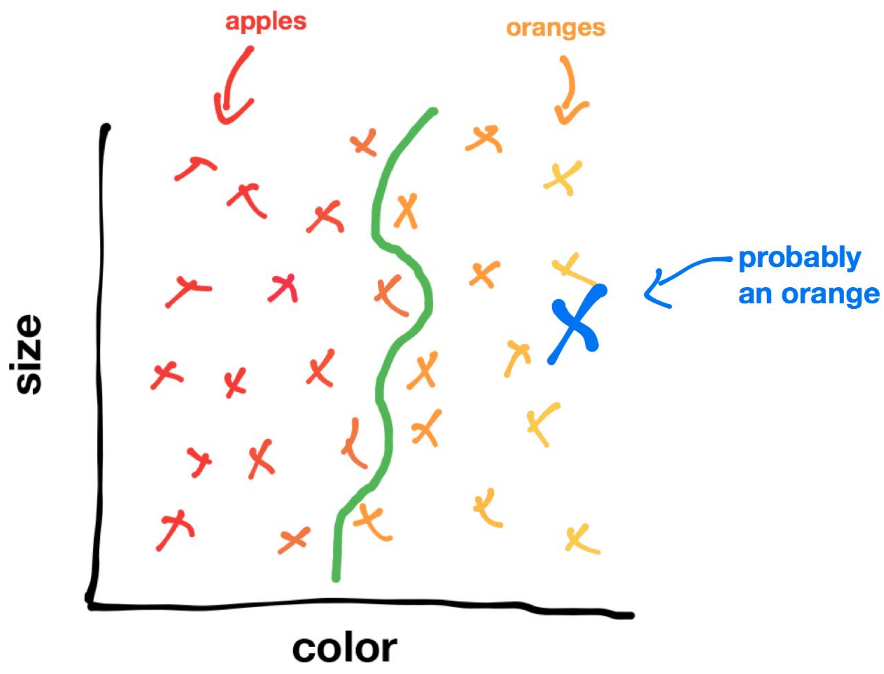
They are the most popular and best performing methods in data mining

Some examples: Random Forest, Adaboost, XGBoost, etc…

Ensembles

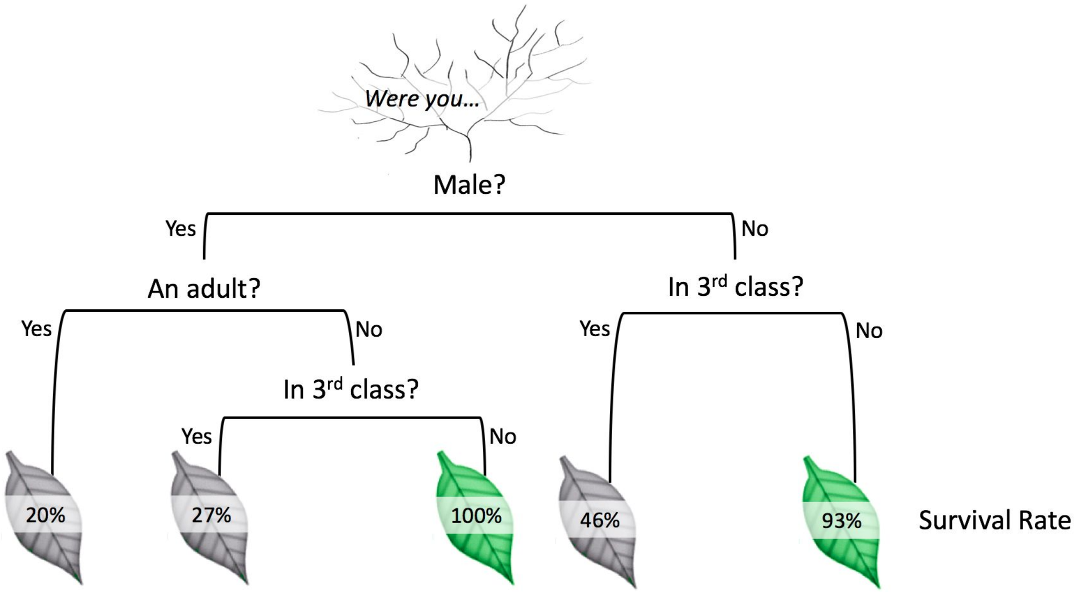
Classification

Size	Color	Label
0.5	1	Apple
0.9	2	Orange
0.6	1	Apple
0.8	2	?

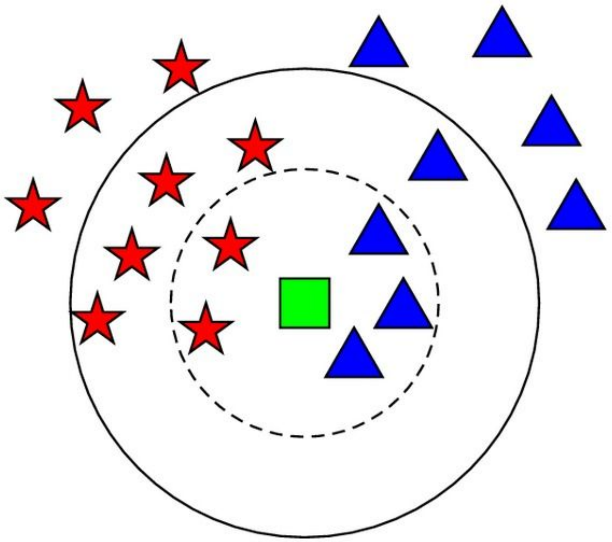


Ensembles

Decision Trees



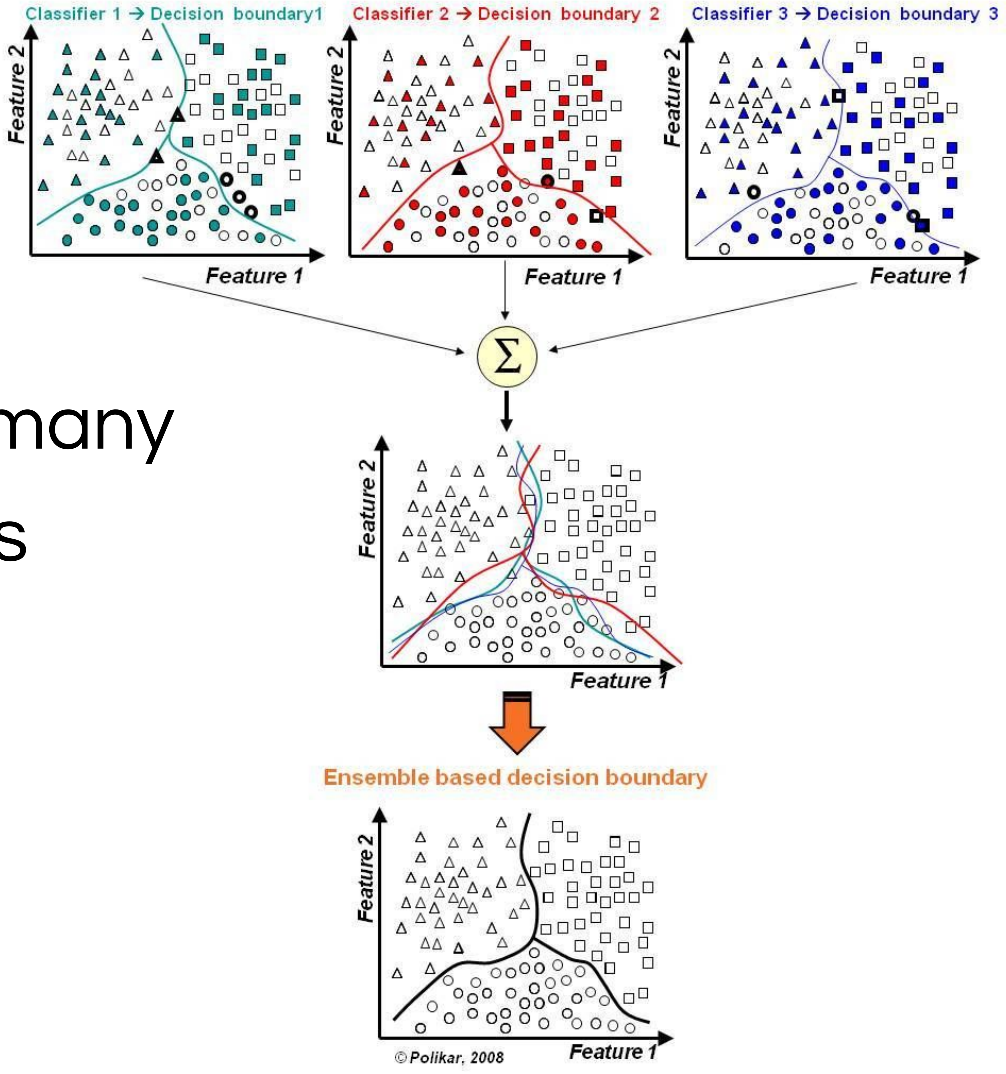
KNN



Ensembles

Combination of base classifiers

Correct errors across many diverse base classifiers



Ensembles

Diversity

Through small changes in input data, diverse classifiers are created and better ensembles are obtained

Input data or classifiers

Ensembles

Bagging trains each model in the ensemble using a randomly drawn subset **with replacement** of the training set

Number of data points in each sample: 63.2%

Boosting incrementally builds an ensemble by training each new model emphasizing the training instances that previous models mis-classified

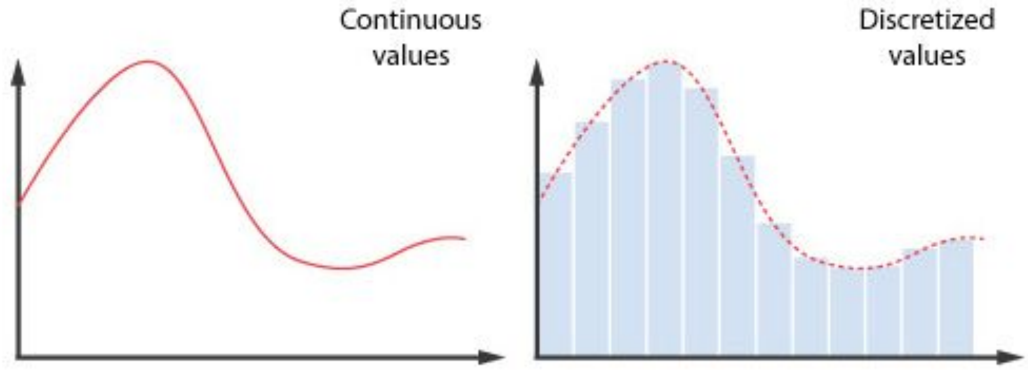
Overfit!

- Distributed and Scalable Ensemble for Big Data
- Focused on diversity
- Inspired by RPRD Ensemble
- Principal Components Analysis (PCA) and Random Discretization (RD)
- Integrated in Spark's MLlib

García-Gil, D., Ramírez-Gallego, S., García, S., & Herrera, F., Principal Components Analysis Random Discretization Ensemble for Scalable Big Data, 2018, Knowledge-Based Systems, doi.org/10.1016/j.knosys.2018.03.012

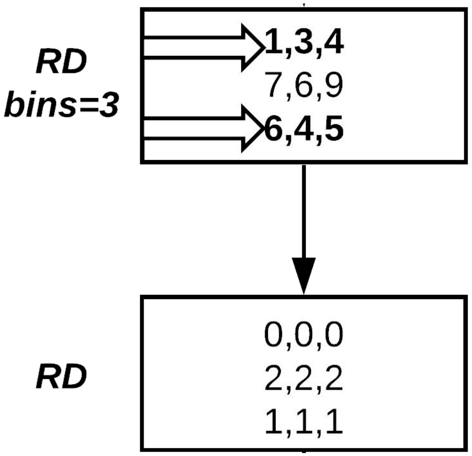
Discretization

It Is the process of transferring continuous variables into discrete ones



Random Discretization

Selects *s*-1 random instances each iteration to create *s* categories



Creates *s* categories

Discretizes the data using those categories

Random Projection

The original *m*-dimensional data is projected to a *d*-dimensional ($d << m$) subspace through the origin, using a random $d \times m$ matrix R whose columns have unit lengths, and whose elements $r_{i,j}$ are often Gaussian distributed.

$$\boldsymbol{X}_{d\times N}^{RP} = \boldsymbol{R}_{d\times m}\boldsymbol{X}_{m\times N}$$

RP Problems

- As the projected dimension is decreased, as it drops below log *k*, RP suffers a gradual degradation in performance

- RP is highly unstable - different random projections may lead to radically different results

Random PCA

More informative method than RP

PCA always offers the same results for a given *k*

Solution: random *k*

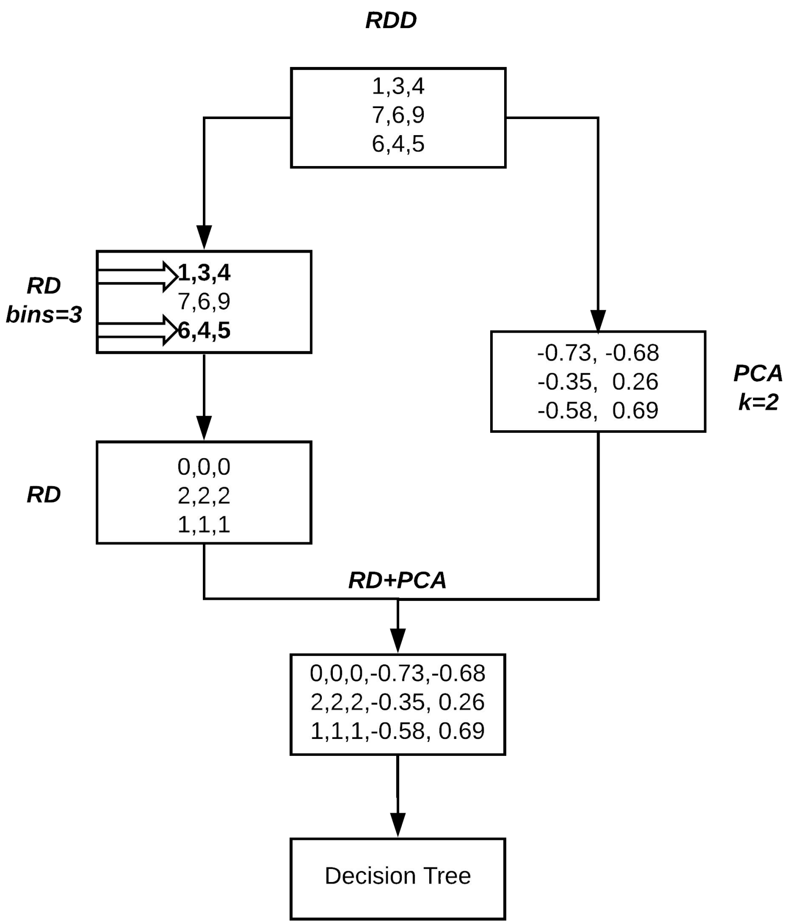
k : [1, m-1] (m number of features)

PCARD

Performs RD and "random" PCA

Joins the results to create more informative data

Learns a Decision Tree



PCARD Datasets

Experimental Framework

Table 1: Summary Description for Classification Datasets

Dataset	Instances	Atts.	Total	CL	Size (GB)
poker	1,025,010	11	11,275,110	10	0.023
SUSY	5,000,000	18	90,000,000	2	2.23
HIGGS	11,000,000	28	308,000,000	2	7.39
epsilon	400,000	2,000	800,000,000	2	14.16
ECBDL14	65,003,913	631	39,847,398,669	2	123.76

PCARD Accuracy

Table 2: RD vs PCA vs PCARDE Test Accuracy using a

Dataset	Trees	RD	PCA
Poker	10	54.73(±0.43)	54.68(±0.24)
	50	54.76(±0.49)	54.81(±0.13)
	100	54.73(±0.28)	54.76(±0.28)
SUSY	10	78.00(±0.09)	75.30(±0.20)
	50	78.26(±0.04)	74.97(±0.08)
	100	78.31(±0.07)	75.31(±0.34)
HIGGS	10	68.64(±0.25)	60.10(±2.01)
	50	68.98(±0.15)	60.44(±0.76)
	100	69.17(±0.12)	60.81(±0.25)
epsilon	10	68.78(±0.39)	78.14(±0.09)
	50	69.04(±0.19)	78.14(±0.09)
	100	69.22(±0.25)	78.14(±0.09)
ECBDL14 [2]	10	0.1884	0.2400
	50	0.1885	0.2410
	100	0.1880	0.2415

PCARD Accuracy

Table 2: RD vs PCA vs PCARDE Test Accuracy using a Decision Tree

Dataset	Trees	RD	PCA	PCARDE
Poker	10	54.73(±0.43)	54.68(±0.24)	55.07(±0.19)
	50	54.76(±0.49)	54.81(±0.13)	54.92(±0.20)
	100	54.73(±0.28)	54.76(±0.28)	54.97(±0.23)
SUSY	10	78.00(±0.09)	75.30(±0.20)	78.31(±0.07)
	50	78.26(±0.04)	74.97(±0.08)	78.47(±0.09)
	100	78.31(±0.07)	75.31(±0.34)	78.49(±0.03)
HIGGS	10	68.64(±0.25)	60.10(±2.01)	68.75(±0.56)
	50	68.98(±0.15)	60.44(±0.76)	69.28(±0.18)
	100	69.17(±0.12)	60.81(±0.25)	69.35(±0.10)
epsilon	10	68.78(±0.39)	78.14(±0.09)	78.57(±0.37)
	50	69.04(±0.19)	78.14(±0.09)	78.57(±0.25)
	100	69.22(±0.25)	78.14(±0.09)	78.58(±0.27)
ECBDL14 [2]	10	0.1884	0.2400	0.4742
	50	0.1885	0.2410	0.4717
	100	0.1880	0.2415	0.4742

PCARD Accuracy

Table 3: PCARDE vs $\mathcal{X}^2$ RD vs RPRD Test Accuracy using a Decision Tree

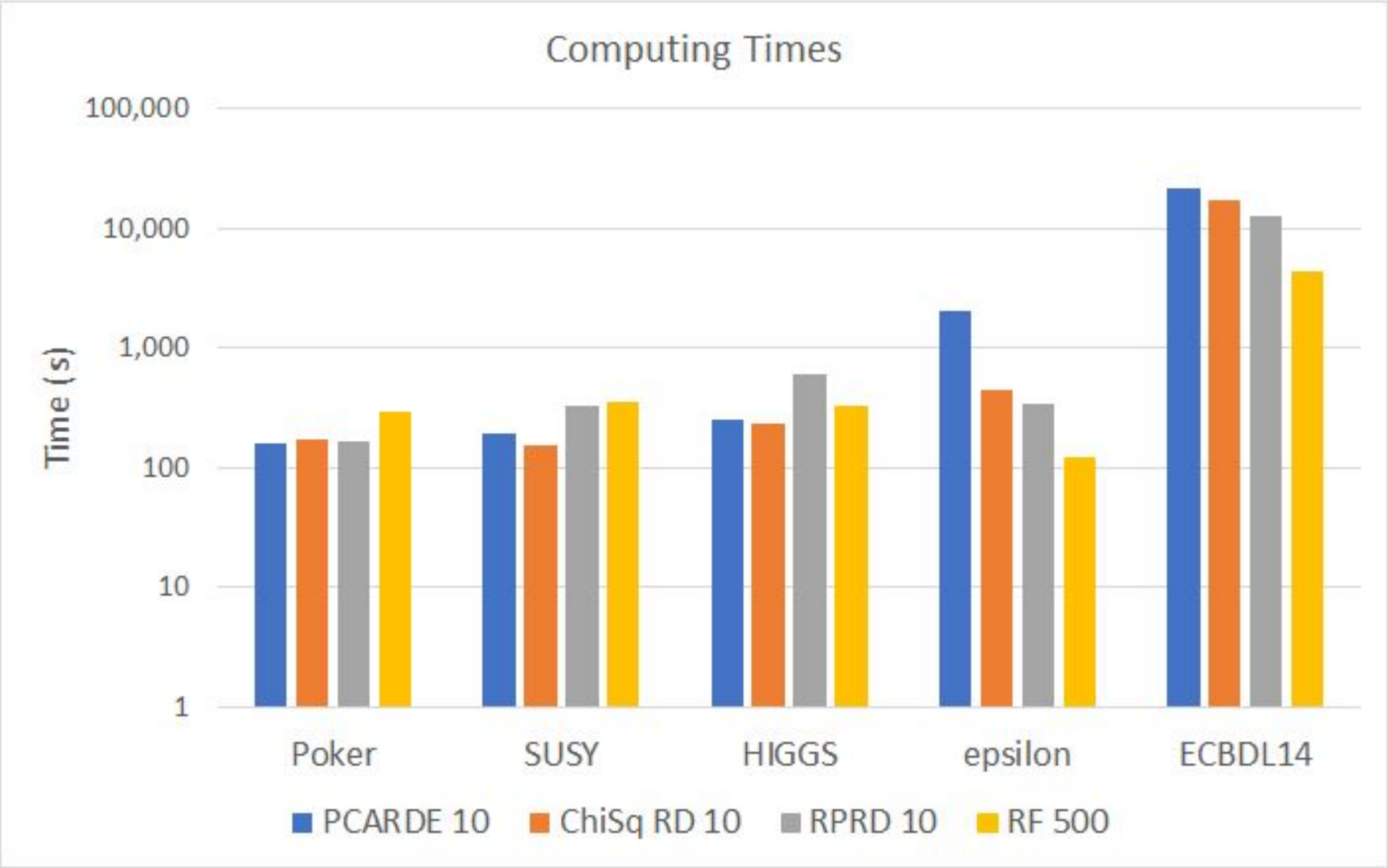
Dataset	Trees	PCARDE	$\mathcal{X}^2$ RD	RPRD
poker	10	55.07(±0.19)	54.72(±0.13)	53.84(±0.26)
	50	54.92(±0.20)	54.64(±0.26)	53.82(±0.25)
	100	54.97(±0.23)	54.70(±0.21)	53.82(±0.07)
SUSY	10	78.31(±0.07)	77.43(±0.15)	78.19(±0.05)
	50	78.47(±0.09)	77.57(±0.18)	78.28(±0.09)
	100	78.49(±0.03)	77.65(±0.17)	78.35(±0.08)
HIGGS	10	68.75(±0.56)	68.48(±0.14)	68.36(±0.09)
	50	69.28(±0.18)	69.06(±0.06)	69.01(±0.13)
	100	69.35(±0.10)	69.18(±0.06)	69.22(±0.17)
epsilon	10	78.57(±0.37)	64.60(±1.33)	68.64(±0.33)
	50	78.57(±0.25)	66.35(±0.34)	69.10(±0.27)
	100	78.58(±0.27)	66.05(±0.80)	69.31(±0.29)
ECBDL14[2]	10	0.4742	0.4512	0.4735
	50	0.4714	0.4524	0.4775
	100	0.4742	0.4519	0.4757

PCARD Accuracy

Table 5: PCARDE vs Random Forest Test Accuracy

Dataset	PCARDE	RF 200	RF 500
poker	55.07(±0.19)	51.56(±0.98)	51.61 (±0.97)
SUSY	78.31(±0.07)	77.73(±0.04)	77.76(±0.07)
HIGGS	68.75(±0.56)	67.98(±0.12)	67.94(±0.13)
epsilon	78.57(±0.37)	73.24(±0.32)	73.41(±0.22)
ECBDL14[2]	0.4742	0.4642	0.4634

PCARD Times


Computing Times
100,000
10,000
1,000
100
10
1
Time (s)
Poker
SUSY
HIGGS
epsilon
ECBDL14
PCARDE 10
ChiSq RD 10
RPRD 10
RF 500

PCARD Times

Table 6: Learning Time Values in Seconds

Dataset	PCARDE 10	$\mathcal{X}^2$ RD 10	RPRD 10	RF 500
poker	159	175	169	294
SUSY	193	151	328	351
HIGGS	248	234	604	325
epsilon	2,048	441	338	124
ECBDL14	22,093	17,280	12,607	4,460

Table 7: Prediction Time Values in Microseconds

Dataset	PCARDE 10	$\mathcal{X}^2$ RD 10	RPRD 10	RF 500
poker	63.41	99.51	78.05	82.93
SUSY	34.00	60.00	41.00	44.00
HIGGS	16.36	30.45	23.18	14.55
epsilon	2,350	412.50	325.00	50.00
ECBDL14	148.97	245.37	214.14	13.10

PCARD

Conclusions:

- PCARD is able to work with huges datasets
- Very stable method with just 10 trees
- Outperforms RPRD and Random Forest
- High computational cost for datasets with lots of features

Outline

- Ensembles
- **Apache Spark**
- MLlib

Apache Spark

RDDs

- Immutable collection of objects
- Partitioned and distributed across a set of machines
- Can be rebuilt if a partition is lost
- Can be cached in-memory to reuse

Apache Spark

- **Transformations**: set of operations of an RDD that define how its data should be transformed
- **Actions**: Applies all transformations on RDD (if has) and then performs the action to obtain results

Transformations	Actions
map (func)	reduce(func)
flatMap(func)	collect()
filter(func)	count()
groupByKey()	first()
reduceByKey(func)	take(n)
mapValues(func)	saveAsTextFile(path)
sample(...)	countByKey()
union(other)	foreach(func)
distinct()	...
sortByKey()	
...	

Outline

- Ensembles
- Apache Spark
- **MLlib**

MLlib

- Decision Trees
  - Scales linearly

- Ensembles of Decision Trees
  - Random Forests
  - Gradient-Boosted Trees

MLlib

Random Forest vs GBT

- GBTs train one tree at a time, so they can take longer to train than random forests. Random Forests can train multiple trees in parallel
- On the other hand, it is often reasonable to use smaller (shallower) trees with GBTs than with Random Forests, and training smaller trees takes less time
- Random Forests can be less prone to overfitting. Training more trees in a Random Forest reduces the likelihood of overfitting, but training more trees with GBTs increases the likelihood of overfitting
- Random Forests can be easier to tune since performance improves monotonically with the number of trees (whereas performance can start to decrease for GBTs if the number of trees grows too large).

In short, both algorithms can be effective, and the choice should be based on the particular dataset

Spark Packages

Available in Spark Packages:

https://spark-packages.org/package/djgg/PCARD

PCARD

PCARD ensemble method. Ensemble of decision trees based on Random Discretization and Principal Components Analysis.

@djgg / Latest release: 1.3 (2018-04-05) / Apache-2.0 / ★★★★★ (3)

1 machine learning | 1 ensemble | 1 mllib

Ensembles in Apache Spark

Big Data II

Diego García (djgarcia@decsai.ugr.es) 10/04/2018