Dimensionality Reduction Methods in Large Scale Data Analytics Libraries

Jing Li, **Zoltan** Andras Lux

23,06,2017

Agenda

- **Introduction**
- **Algorithms**
  - sPCA
  - TDA
  - (t)-SNE
  - LSH
- **Sampling**
- **Big Data Libraries**
- **Summary**
- **Demo**
  - PCA demo with Spark

Intro - The curse of dimensionality

- Real data usually have thousands, or millions of dimensions
  - E.g., web documents, where the dimensionality is the vocabulary of words
  - Facebook graph, where the dimensionality is the number of users
- Huge number of dimensions causes problems
  - Data becomes very sparse, some algorithms become meaningless
  - The complexity of AI algorithms depends on the dimensionality and they become infeasible.
  - E.g., K-means Clustering: everything is "far away"

Intro - Why

- Redundancy reduction and intrinsic structure discovery
- Intrinsic structure discovery
- Removal of irrelevant and noisy features
- Feature extraction
- Visualization purpose
- Computation and Machine learning perspective
- …

Intro - Dimensionality Reduction

- The main idea: Represent data in a low-dimensional space
- Project the d-dimensional points in a k-dimensional space so that:
  - k << d
  - distances are preserved as well as possible

Solve the problem in low dimensions

Intro - Dimensionality Reduction

- Visualization
  - Insights into high-dimensional structures in the data
- Better generalization
  - Fewer dimensions → less chances of overfitting
- Speeding up learning algorithms
  - Most algorithms scale badly with increasing data dimensionality
- Data compression
  - Less storage requirements

Algorithms

- Linear methods
  - Principal Component Analysis (PCA), (scalable)sPCA, onlinePCA
  - Singular Value Decomposition(SVD)
  - Independent Component Analysis(ICA) - Projection Pursuit
  - Metric Multidimensional Scaling(MDS)
  - Topological Data Analysis(TDA)

- Nonlinear methods
  - (t)-Distributed Stochastic Neighbor Embedding (t)-SNE
  - Locality Sensitive Hashing(LSH)
  - Locally Linear Embedding (LLE), Hessian LLE
  - Isomap

Algorithms - Overview

- Linear
  - **PCA**: directions most variance; additional functionalities - noise reduction, ellipse fitting, and solutions for non-full rank eigenproblems
  - **LDA**: discrete label information, the reduced feature vectors are efficient for discriminant (classification).
  - **MDS**: Euclidean space, configuration of points in a target metric space from information about inter-point distances
- Limitations: Effectiveness is limited by its global linearity; only characterize *linear* subspaces (manifolds)
- Nonlinear:
  - **LLE**: exploits the relationships between each point and its neighbors, neighborhood preservation
  - **ISOMAP**： pairwise geodesic distance, neighborhood graph, K-nearest neighbors
- Provide ability to discover the latent space which is nonlinearly embedded in the original feature space

Dimensionality Reduction A Short Tutorial
Ali Ghodsi Department of Statistics and Actuarial Science University of Waterloo Waterloo, Ontario, Canada, 2006

Algorithms - (scalable) **sPCA**

- **Method:**
  - PCA: a latent variable model that seeks a linear relation between a D-dimensional observed data vector **y** and a d-dimensional latent variable **x**.

$$\mathbf{y} = \mathbf{C} * \mathbf{x} + \mu + \varepsilon$$

  - Given N observations ***{y}$^{n}{}_{1}$***as the input data, the log likelihood is given by:

$$L(\{\mathbf{y}\}) = \sum_{r=1}^{N} \ln\{p(\mathbf{y})\}$$

  - Main idea: MLE of C is obtained by optimizing ***argCmaxL({y})***.
    The main idea of probabilistic PCA is that the MLE solution is the solution of PCA.
    Expectation Maximization algorithm is adopted to solve MLE problem.

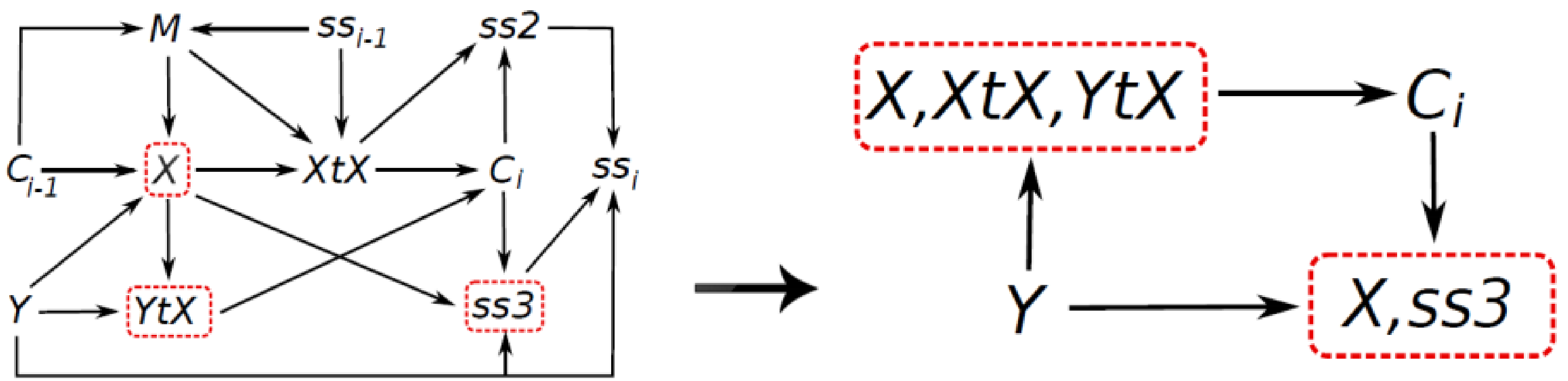
Algorithms - (scalable) **sPCA**

Algorithm 1 PPCA (Matrix Y, int N, int D, int d)

$C = normrnd(D, d)$
$ss = normrnd(1, 1)$
$Ym = columnMean(Y)$
$Yc = Y - Ym$
while not STOP_CONDITION **do**
    $M = C' * C + ss * I$
    $X = Yc * C * M^{-1}$
    $XtX = X' * X + ss * M^{-1}$
    $YtX = Yc' * X$
    $C = YtX / XtX$
    $ss2 = trace(XtX * C' * C)$
    $ss3 = \sum_{n=1}^{N} X_n * C' * Yc'_n$
    $ss = (||Yc||_F^2 + ss2 - 2 * ss3)/N/D$
end while

Algorithms - (scalable) **sPCA**

- From PPCA to sPCA



- Propagate mean to leverage sparsity, and keep original matrix Y and mean vector Ym in two separate data structures

$$Y_C * C = (Y - Ym) * C = Y * C - Ym * C$$

- Minimize the intermediate data. Store vectors and small matrices locally and trade intermediate data footprint with redundant computation.
- Make matrix multiplication more efficient.

$$(A_T * B) = \sum_{i=1}^{D} (A_i)^T * B_i$$

Algorithms - (scalable) **sPCA**

Dataset	Size	sPCA-Spark	MLlib-PCA	sPCA-MapReduce	Mahout-PCA
Tweets	$1.26B \times 2K$	708	822	3,900	29,160
	$1.26B \times 6K$	1,260	2,196	10,080	97,920
	$1.26B \times 71.5K$	5,940	Fail	16,200	430,200
Bio-Text	$8.2M \times 2K$	48	102	1,050	2,280
	$8.2M \times 10K$	114	Fail	1,290	6,240
	$8.2M \times 14K$	516	Fail	1,740	8,580
Diabetes	$353 \times 2K$	20	55	540	720
	$353 \times 10K$	30	Fail	720	1,680
	$353 \times 65.7K$	156	Fail	960	3,300
Images	$160M \times 128$	7,800	660	12,600	117,700

Comparison of running time (in sec) for sPCA on both Spark (sPCA-Spark) and MapReduce (sPCA-MapReduce) against the closest counterparts on Spark (MLlib-PCA) and MapReduce (Mahout-PCA).

Algorithms - TDA

- Large, complex and high-dimensional data sets
- Mathematical interpretation - homology groups
  - Nodes
  - Edges
- Main tasks
  - The measurement of shape
    - “measure” shape
  - The representation of shape
    - find compressed combinatorial representations of shape and analyze the degree to which these representations are faithful to the shape

Algorithms - TDA

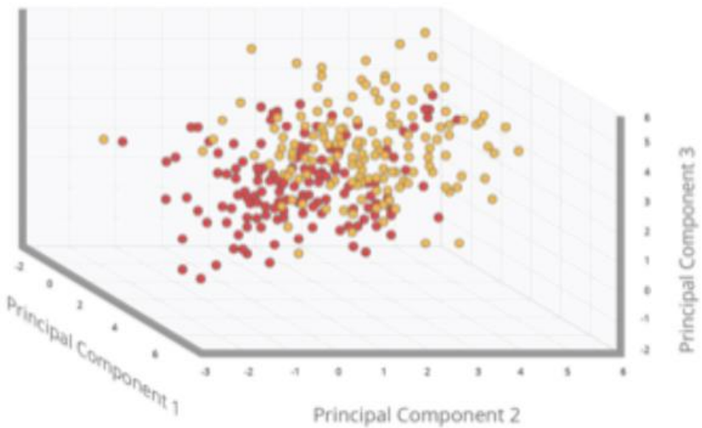
- Properties
  - COORDINATE INVARIANCE
  - DEFORMATION INVARIANCE
  - COMPRESSED REPRESENTATIONS

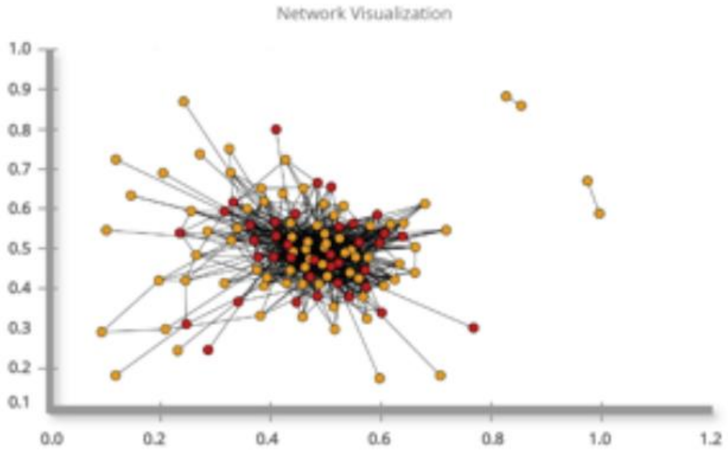
Algorithms - TDA

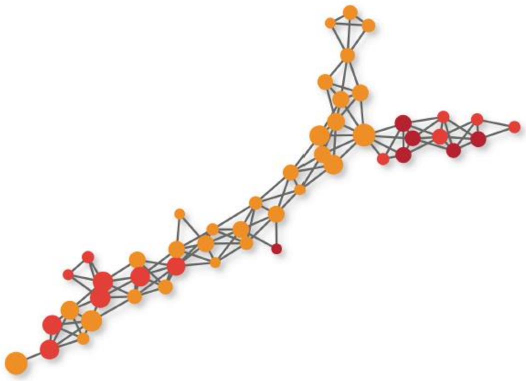
- Topological Networks
  - A (geographic) map of all the points in the data set
  - Representation
    - Each node corresponds to multiple data points
    - #node<<#data points
  - Visualization
  - Interactive model
  - Easy to interrogate

Algorithms - TDA

- A framework for Machine Learning
  - “Shape has meaning”
  - Interrogate ML outputs – highlights high value segments of the data







  - In conjunction with ML – to understand the “shape” of complex data sets.

Algorithms – t-SNE

- Extended version of SNE
- Instead of the Gaussian, t-SNE uses a heavy tailed distribution
  - This is useful for preserving large dissimilarities between distant points

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}$$

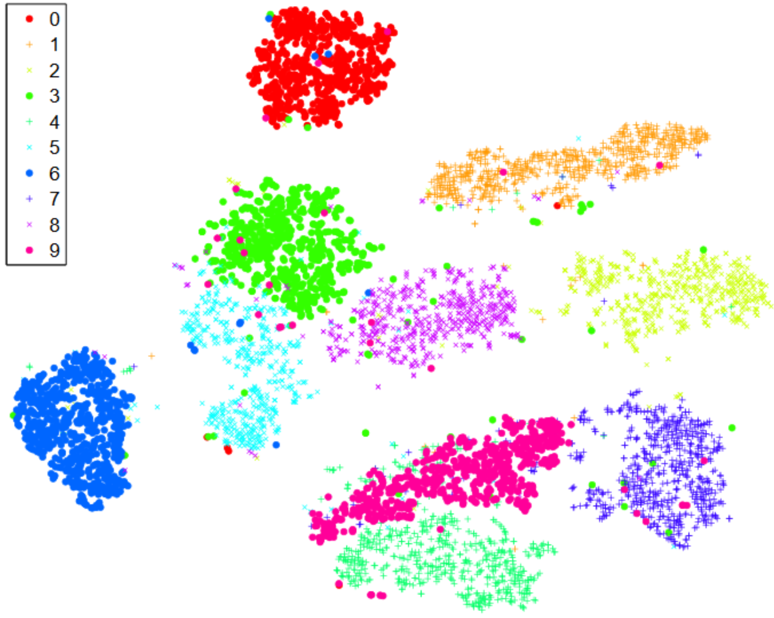
Algorithms – t-SNE

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity *Perp* (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=*1* **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
  end
end

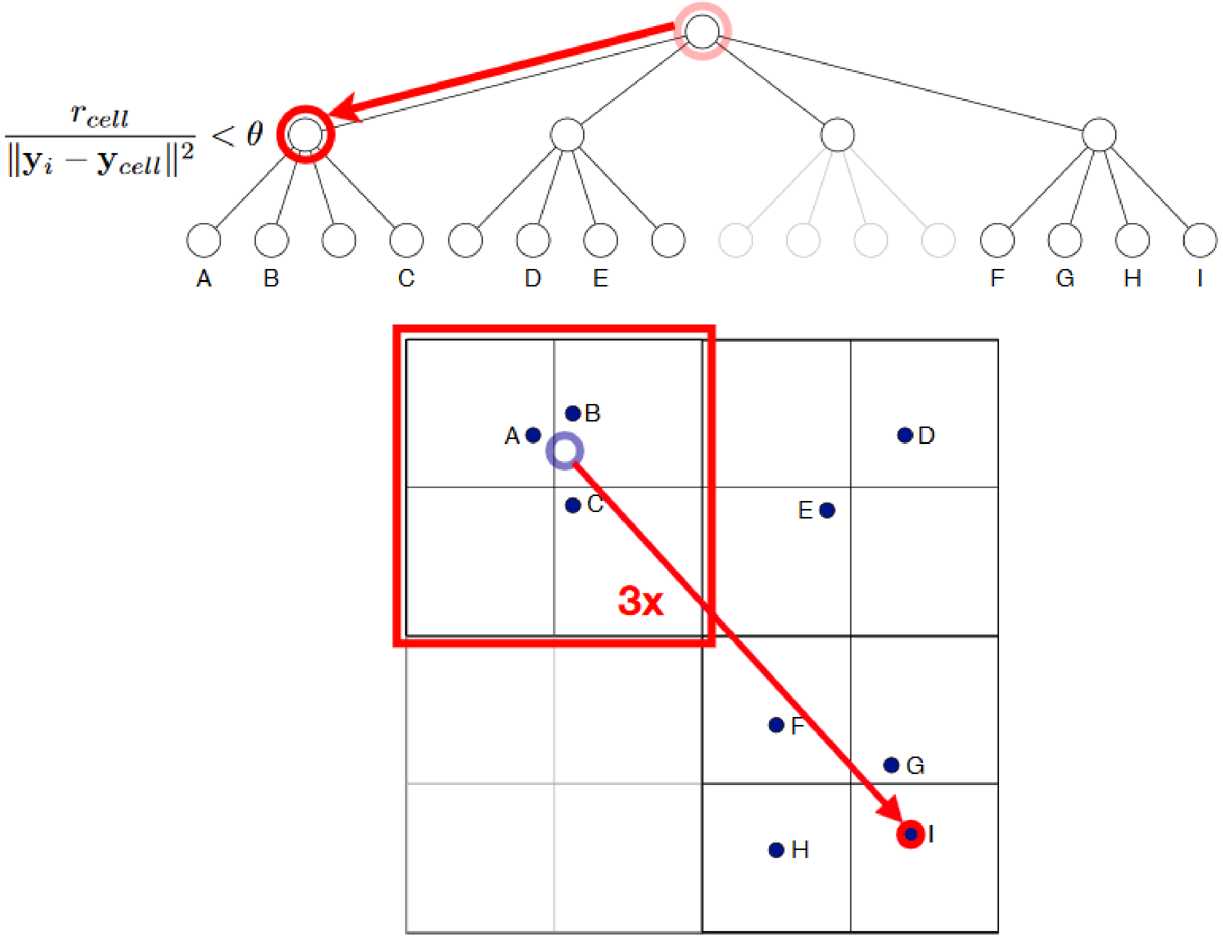
Algorithms – t-SNE

- MNIST visualization by t-SNE

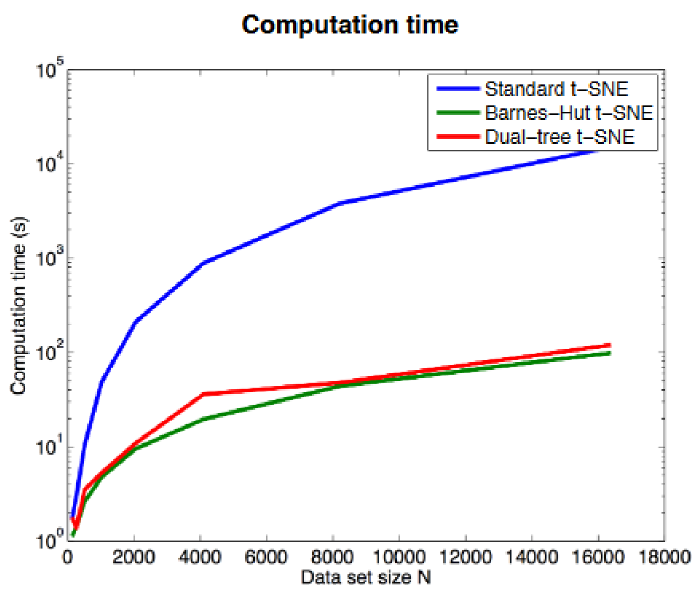


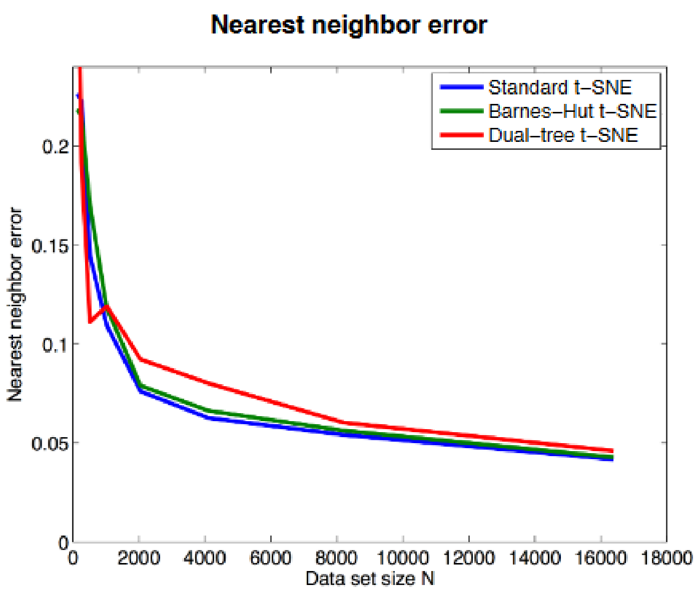
(a) Visualization by t-SNE.

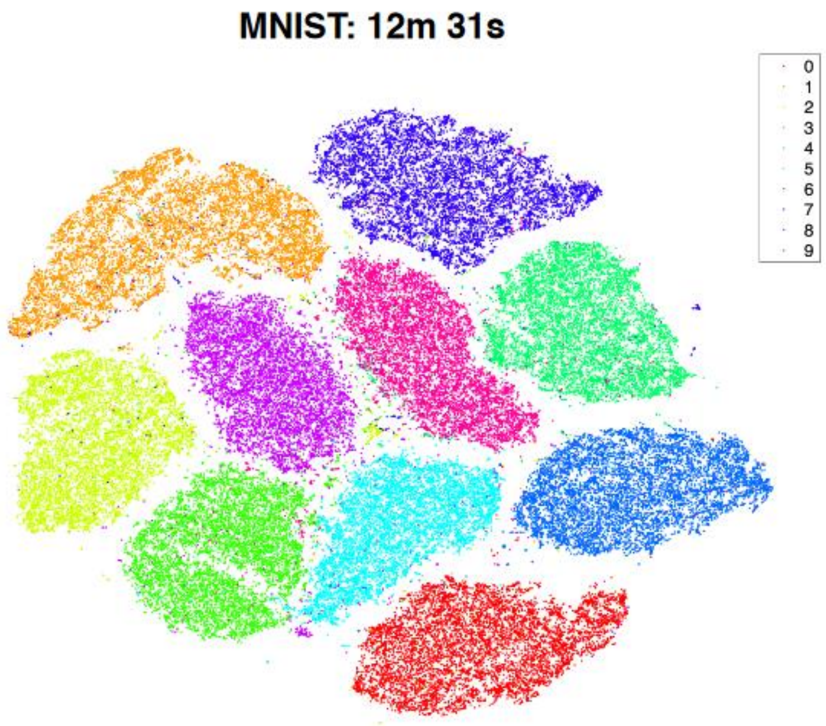
Algorithms – t-SNE



Algorithms – t-SNE





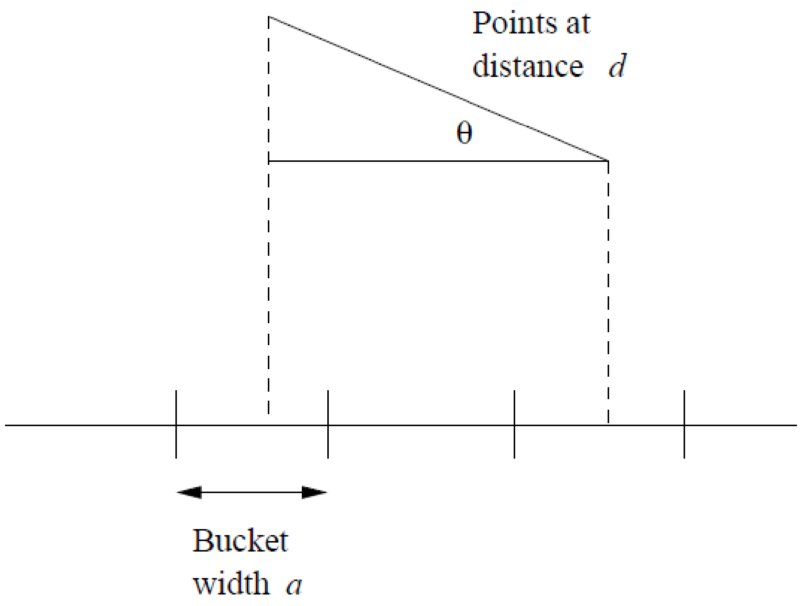


Algorithms – Locality Sensitive Hashing

- We want to map the points to a smaller space in a way such that distances between pairs of points are nearly preserved
- Also applied for Nearest Neighbor Search (NNS)
- Only approximate solution with high probability of correctness

Algorithms – LSH in Euclidean Space

- Start with line ***f*** in ***F***
- Divide ***f*** into segments of length ***a***
- Segments are buckets
- Project point on the line to find its bucket



Algorithms – LSH for Minhash signatures

- Minhash
  - To Minhash a set represented by a column of the characteristic matrix, pick a permutation of the rows. The Minhash value of any column is the number of the first row, in the permuted order, in which the column has a 1
  - The probability that the Minhash function for a random permutation of rows produces the same value for two sets equals the Jaccard similarity of those sets
  - Replace matrix M by its signature Matrix
- Hash items several times
- Items for any hash in the same buckets are considered candidates

Algorithms – Amplifying a Locality-Sensitive Family

Let $d_1 < d_2$ be two distances according to some distance measure d. A family $\mathbf{F}$ of functions is said to be (d_1, d_2, p_1, p_2)-*sensitive* if for every f in $\mathbf{F}$:

1. If $d(x, y) \leq d_1$, then the probability that $f(x) = f(y)$ is at least p_1.

2. If $d(x, y) \geq d_2$, then the probability that $f(x) = f(y)$ is at most p_2.

- Given a ***(d1, d2, p1, p2)***-sensitive family F. We can construct a new family F′ by the AND-construction on F with r members
- F′ is a ***(d1, d2, $(p1)^r$, $(p2)^r$)***-sensitive family
- OR-construction: turns a ***(d1, d2, p1, p2)***-sensitive family ***F*** into a ***(d1, d2, $1 - (1 - p1)^b$, $1 - (1 - p2)^b$)*** - sensitive family ***F′***

Algorithms – LSH and Distance Measures

- Euclidean Distance
- Cosine Distance
- Hamming Distance
- Jaccard Distance

Algorithms - Summary

- sPCA
  - Pro: scalable, support large datasets on distributed clusters
  - Con: the accuracy depends on the number of iterations， many intermediate variables
- TDA
  - Pro: simple and intuitive, very good for visualization
  - Con: high sensitivity of homology
- t-SNE
  - Pro: very good for visualization, tries to preserve pairwise distances
  - Con: not implemented in big data libraries, generally for visualization, no global optimum guranteed, needs iterative computation
- LSH
  - Pro: scalable and fast
  - Con: approximate solution

Sampling

- Sampling: obtain a small sample S to represent the whole data set N. Choose a representative subset of the data

- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- To reduce the number of instances submitted to the DM algorithm.
- To support the selection of only those cases in which the response is relatively homogeneous.
- To assist regarding the balance of data and occurrence of rare events.

Sampling

- Forms of data sampling (T → data set, N → nº examples):

- **Simple random sample without replacement (SRSWOR) of size s**: This is created by drawing **s** of the **N** tuples from **T** (**s** < **N**), where the probability of drawing any tuple in **T** is 1/**N.**
- **Simple random sample with replacement (SRSWR) of size s**: Similar to SRSWOR, except that each time a tuple is drawn from **T** , it is recorded and replaced.

Sampling

- **Balanced sample:** The sample is designed according to a target variable and is forced to have a certain composition according to a predefined criterion.
- **Cluster sample:** If the tuples in T are grouped into G mutually disjointed clusters, then an SRS of s clusters can be obtained, where $s < G$.
- **Stratified sample:** If T is divided into mutually disjointed parts called strata, a stratified sample of T is generated by obtaining an SRS at each stratum

Big Data Libraries and Dimensionality Reduction

Library	Algorithm	Languages	Remarks
Mahout			Spark backend is stable, MapReduce is deprecated
	SVD	MapReduce,Spark,H2O,Flink	
	Lanczos Algorithm	MapReduce	
	Stochastic SVD	MapReduce,Spark,H2O,Flink	
	QR Decomposition	MapReduce,Spark,H2O,Flink	
	PCA (with SVD)	MapReduce,Spark,H2O,Flink	
Spark ML			Dataframe based
	PCA	Java, Scala, Python	
	LSH (Jaccard and Euclidean Distance)	Java, Scala, Python	
Spark MLlib			RDD bases, older than Spark ML
	PCA	Java, Scala	
	SVD	Java, Scala	
H2O			
	PCA	Java, Scala, Python, R	
Apache SINGA			Still in incubator, capable of utilizing GPUs
	Autoencoders	C/C++, configuration file	

Summary

- Several Algorithms Exists, the most widely known and implemented method is PCA/SVD
- Algorithms exist for several types of data
- Scalability has to be addressed
- Some information loss is inevitable through dimensionality reduction

DEMO

PCA in Spark

References

i. **Dimensionality Reduction,** Wei-Lun Chao, Graduate Institute of Communication Engineering, National Taiwan University

ii. **Dimensionality Reduction A Short Tutorial**, Ali Ghodsi, Department of Statistics and Actuarial Science, University of Waterloo,Waterloo, Ontario, Canada, 2006

iii. **Dimensionality Reduction: A Comparative Review,** Laurens van der Maaten, Eric Postma, Jaap van den Herik TiCC, Tilburg University

iv. **Big Data Reduction Methods: A Survey**, Muhammad Habib ur Rehman, Chee Sun Liew, Assad Abbas, Prem Prakash Jayaraman,Teh Ying Wah, Samee U. Khan

v. **Frontiers in Massive Data Analysis**

vi. **sPCA: Scalable Principal Component Analysis for Big Data on Distributed Platforms,** Tarek Elgamal, Maysam Yabandeh, Ashraf Aboulnaga, Waleed Mustafa, Mohamed Hefeeda, Qatar Computing Research Institute, Twitter, NTG Clarity

vii. **AYASDI, Topology & Topological Data Analysis**

viii. **Topological Data Analysis: A Framework for Machine Learning**

References

i. **Visualizing High-Dimensional Data Using t-SNE**, L.J.P. van der Maaten, G.E. Hinton, Journal of Machine Learning Research 9(Nov):2579-2605, 2008.
ii. **Accelerating t-SNE using Tree-Based Algorithms,** L.J.P. van der Maaten. *Journal of Machine Learning Research* 15(Oct):3221-3245, 2014.
iii. **MMDS**, Chapter 11 Dimensionality Reduction
iv. Spark Machine Learning Library (MLlib)
v. Apache Mahout
vi. Apache SINGA
vii. H2O