Random Projection Forests

Donghui Yan

Mathematics and Data Science@UMassD

Center of Mathematical Sciences and Applications,
Harvard University

January 18, 2019

Outline

- Introduction to *rpForests*
- k-nearest neighbor (kNN) search
- Scoring of tissue microarray (TMA) images
- Summary

Random projection forests (*rpForests*)

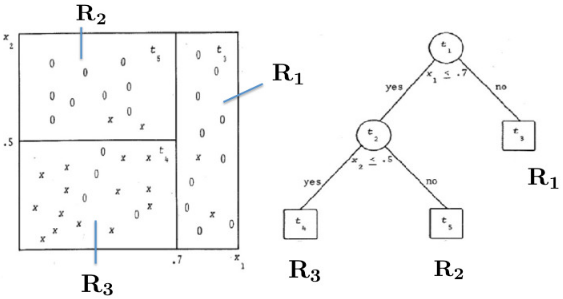
- Ensemble of trees constructed recursively on random projections
- Combines power of ensemble methods and flexibility of trees
- Discovers patterns, e.g., locality, useful for various applications.

Tree-based methodology

- A broad class of methods in statistics and data mining
  - e.g., C4.5, CART, QUEST, Random Forests, GBM etc
- Huge impact in many areas
  - Medicine, finance, commerce etc
- Fast computation
  - Computational complexity $O(n \log(n))$ for tree growth
  - $O(\log(n))$ for search or prediction
- Typically decent performance
- Good interpretability
  - Resembles the human decision dichotomy.

Illustration of tree-based methods

- 1^{st} partition along variable x_1: $R_1, (R_2 \cup R_3)$
- 2^{nd} partition along variable x_2: R_2, R_3
- Fitted function: $h(R_1) =$ ‘$0'$, $h(R_2) =$ ‘$0'$, $h(R_3) =$‘x’.

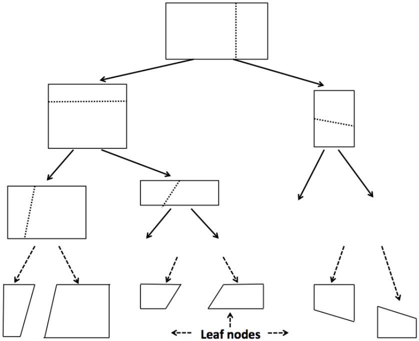


Extensions

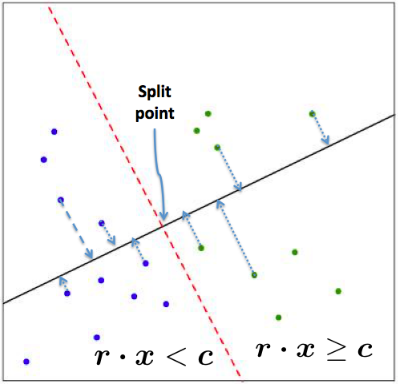
- Existing methods typically require responses for tree growth
  - i.e., in supervised learning mode
  - Classification
    - Node split to optimize entropy or Gini
  - Regression
    - Node split to optimize sum of squared errors
- What if no responses are available?
  - i.e., unsupervised learning mode
  - How to grow the tree then?
    - One strategy is to randomly split the nodes
    - e.g., *random projection trees* (Dasgutpa and Freund 2008).

Random projection trees (rpTrees)

- Instead of optimizing w.r.t. a goodness metric
  - *Randomly* pick a split direction
  - *Randomly* pick a split point along the direction.

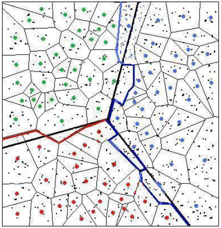


Tree node split by random projection



Random projection trees

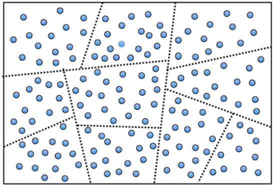
- Does this thing work?
  - Tree growth seems *quite random*
- Theoretical evidence (Dasgupta and Freund 2008)
  - Radius of tree nodes shrinks steadily with depth
  - Automatically adapts to intrinsic dimensionality



- Empirical evidence (Yan, Huang, Jordan 2009)
  - Fast spectral clustering with rpTrees to group data for larger computational units.

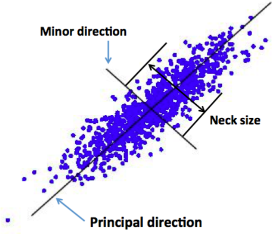
Random projection trees

- rpTrees works in fast spectral clustering
  - Class boundary changes very little despite randomness
- *However*, result may not be stable or satisfactory if
  - Problem depends on pointwise *locality* of data
  - Locality compromised at boundary of leaf nodes.



Random projection forests (*rpForests*)

- Ensemble as an easy way to make rpTrees *great*
  - Locality lost in one tree compensated by others
  - Computationally efficient
    - Easily run on clustered or multicore computers
  - Locality improved *exponentially* by ensemble



Let S be a set of data points with neck size ν. Assume each tree in ensemble $\mathcal{T}$ splits at most J times, and the neck of child nodes shrinks by at most a factor of $0 < \gamma < 1$. Then, the probability that two points A and B will be separated is at most

$$\left(\frac{2d_{AB}}{\pi\nu} \frac{1}{\gamma^{J-2}(1-\gamma)} \right)^{|\mathcal{T}|}.$$

Related work

- Random Forests (RF, Breiman 1999)
- Random projection trees (Dasgupta and Freund 2008)
- Greedy random forest classifier (Biau, Devroye, Lugosi 2015)
- Random projection ensemble classification (Cannings and Samworth 2017)
- Cluster Forests (Yan, Chen and Jordan 2013)
  - Cluster information gathered from many perspectives
  - Random feature pursuits to produce 'good' views of data
  - *Unsupervised extension to RF.*

kNN search

kNN search

Wide applications of kNN

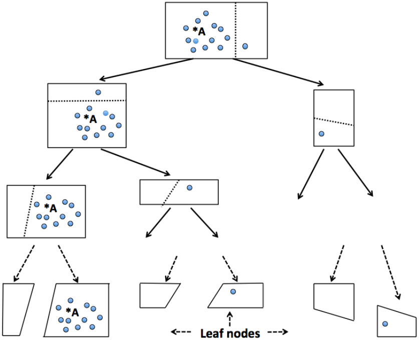
- Data mining
  - Similarity search
- Machine learning
  - To sparsify the Gram matrix for fast computation
- Statistics
  - Nonparametric density estimation
  - kNN-based hypothesis testing
  - Intrinsic dimension estimation
- Anomaly detection.

Challenges and existing work

- Computation $O(n^2)$ in a naive implementation
  - Linear computation necessary for big data
- Existing algorithms
  - Cover tree (Beygelzimer, Kakade and Langford 2006)
    - Slow index building and inefficient use of memory
  - Locality sensitive hashing (Andoni and Indyk 2008)
    - Requires to design hash function
  - (Randomized) k-d trees (Bentley 1975, Hartley 2008)
    - May suffer from dimensionality curse
  - Random projection trees (Dasgupta and Sinha 2015)
    - Needs to route data to multiple leaf nodes
    - Not easy to implement.

kNN by random projection trees

- Local search—kNNs within a tree node
  - Neighboring points tend to be in same leaf node
    - Reduce search from entire data to a tree leaf node
- Potential miss near boundary of leaf nodes.



kNN search by rpForests

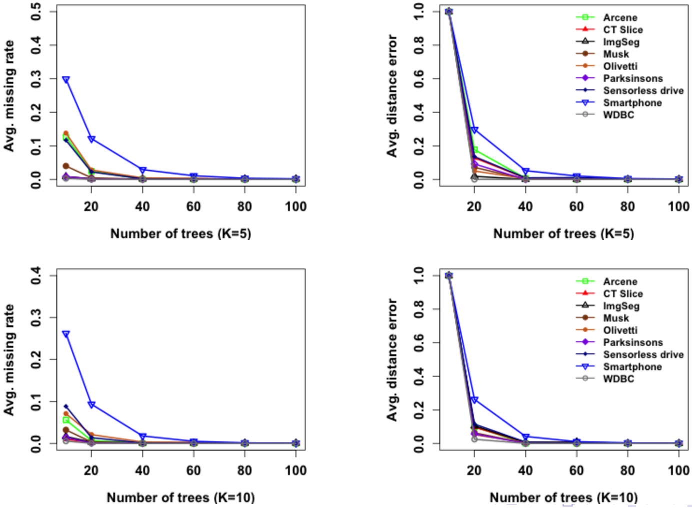
- Search within union of leaf nodes from all trees
  - Miss rate decreases sharply with the ensemble size
    - *Locality* improved exponentially by ensemble
- Crucial observation for *enhancement*
  - The error bound *inversely* proportional to neck size
- Desirable to prevent neck size from becoming too small
  - Avoid cutting data along the minor direction
  - Or, try to split data along its principal direction
  - Algorithmic implementation
    - Sample a few random directions
    - Pick one s.t. the projections have largest variance.

Experiments on accuracy

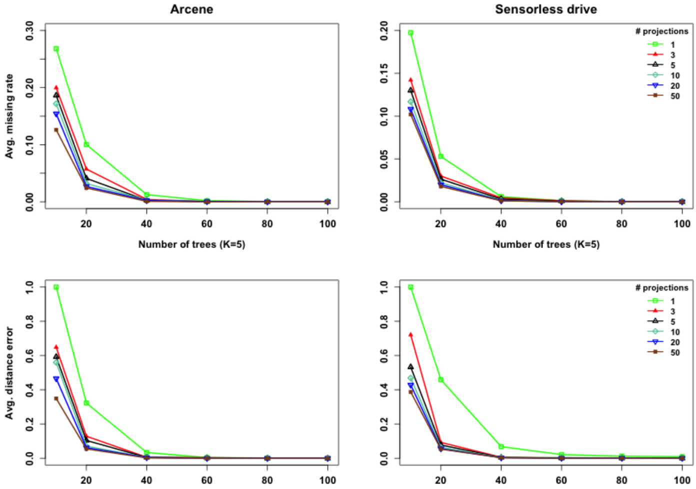
- Two performance metrics
  - Average miss rate out of kNNs
  - Difference between true and computed kNN distances
    - Normalized by true kNN distance.

Dataset	Features	#Instances
Image Segmentation	19	2100
Parkinson's Telemonitoring	20	5815
Wisconsin breast cancer (WDBC)	30	569
Sensorless Drive	49	58509
Musk	166	6598
CT Slice Localization	386	53500
Smartphone Activity	561	7767
Arcene	10000	700
Olivetti Face	10304	400

Experiments on accuracy



Effects of #projections sampled

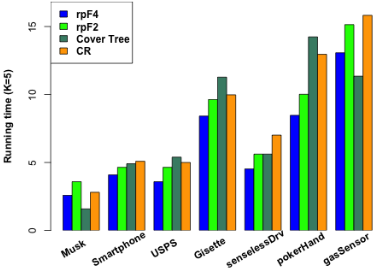


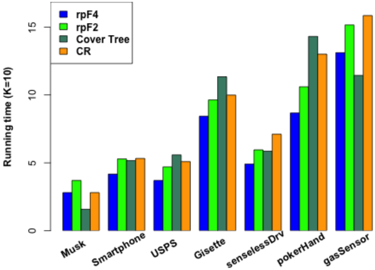
Experiments on running time

- Multicore computers (2- and 4-core).

Dataset	Features	#Instances
Musk	166	6,598
Smartphone	561	7,767
USPS digits	256	11,000
Gisette	5000	12,500
Sensorless Drive	49	58,509
Poker hand	11	1,000,000
Gas sensor array	19	4,178,504

Experiments on running time





Scoring of TMA images

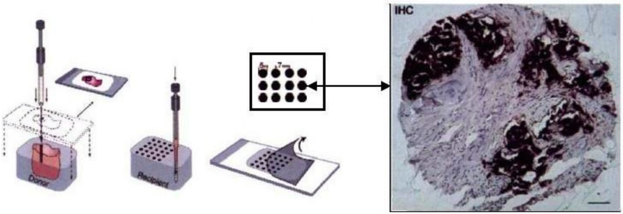
Scoring of TMAs

Tissue microarray images

- Measure tumor-specific protein expression level
- Wide applications
  - Clinical outcome analysis
  - Tumor progression analysis
  - Identification of predictive or prognostic factors
  - Development of new biomarkers
  - Validation of tumor markers (e.g., IHC, FISH etc)
  - Study of genomics and proteomics ("imaging genetics")
    - E.g., analysis of genetic alterations.

Producing TMA images

- Obtain tissue cores from tumor site and store in archive
- Section slices of tissues and mount onto in form of array
- Apply biomarker (stain) and take images.

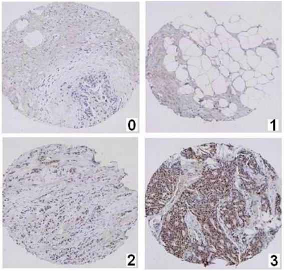


- Each cell in a TMA array $\Longleftrightarrow$ a tissue (image)
  - Each cell in a microarray image $\Longleftrightarrow$ a gene.

The scoring of TMA images

Measure tumor-specific protein expression level

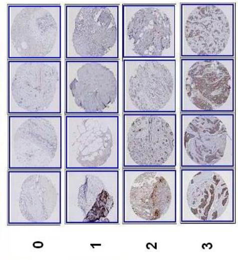
- 0 - definite negative (no staining)
- 1 - ambiguous or weak staining in a minority of tumor cells
- 2 - weak positive (minor dark or major weak nucleus staining)
- 3 - definite positive (majority show dark nucleus staining)



Scoring algorithms

- Manual scoring
  - Variability, subjectivity, and labor-intensive
- Previous algorithms
  - AQUA, ACIS, TMALab, Ariol etc
  - Rely on background subtraction or image segmentation
    - Thresholds for hue, shape and intensity etc
    - Sensitive to variations by noise or illumination etc
    - Image segmentation is often difficult for textured images
  - May require extensive tuning from vendors
- TACOMA (Yan, Wang, Knudsen, Linden and Randolph 2012)
  - Rivals pathologists in accuracy, reproducibility and efficiency.

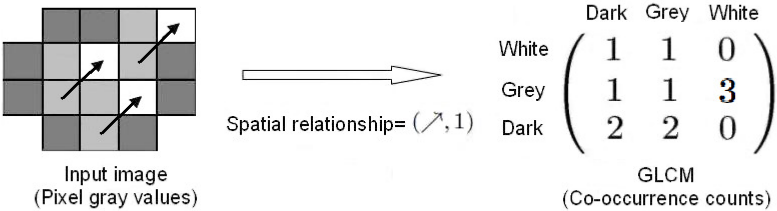
Challenges and idea of TACOMA



- TMA images highly heterogenous
  - Staining patterns not localized in position, shape, or size
- However, image statistics show stable patterns
  - Grey level co-occurrence matrix (GLCM) statistics
- Selective image patches to incorporate pathologists knowledge
- GLCM's input to Random Forests (RF).

GLCM (Grey Level Co-occurrence Matrix)

- Each TMA image $\Longrightarrow$ a GLCM
  - GLCM as features for an image in classification
- GLCM as histogram of gray values of neighboring pixels with a given spatial relationship.



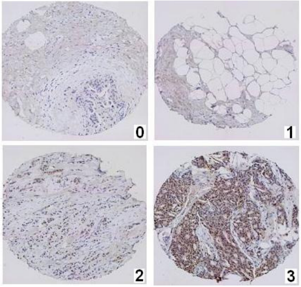
Input image
(Pixel gray values)

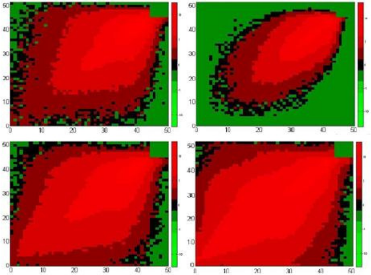
Spatial relationship= $(\nearrow, 1)$

$$\begin{array}{c|ccc} & \text{Dark} & \text{Grey} & \text{White} \\ \hline \text{White} & 1 & 1 & 0 \\ \text{Grey} & 1 & 1 & 3 \\ \text{Dark} & 2 & 2 & 0 \end{array}$$

GLCM
(Co-occurrence counts)

GLCM of TMA images (heat map in log scale)





How to advance the state-of-the-art?

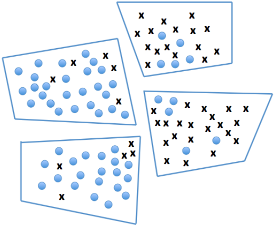
- TMA images inherently small sample
  - Unlike natural images on which DL has huge success
  - TMA images scored by tumor and biomarker type
  - Expensive or forbidden to acquire in large quantity
- Inspiration from recent success in deep learning
  - As a triumph in representation learning
    - Rather than advance in classification technology
- *Deep* features derived from existing ones by computation
  - ♠ *deepTACOMA* algorithm.

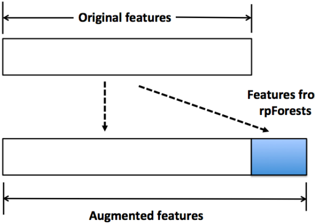
Ideas to look for deep features

- Driven by specific challenges in TMA scoring
  - Labels (scores) are noisy, affected by many variations
    - Illumination of the display device for the image
    - Variations among scorers
    - Status of a scorer and adjacent images in the sequence
- How to reduce the effect of label noise?
  - Solution: look for features that capture locality of points
    - Intuition: *similar images should have same labels*
  - *Regularization* effect in statistics.

Deep features from *rpForests*

- Membership of points in tree leaf nodes by *rpForests*
  - *Cluster* assumption in semi-supervised learning
    - Borrow info from labeled instances (higher quality ones here)
  - Also helps deal with heterogeneity
    - Further signals classifier to build submodels when necessary.





Experiments

- TMA images corresponding to ER (Estrogen Receptor)
  - Available at the Stanford TMA repository (*tma.stanford.edu*)
  - Totally 695 images with 50%-50% for training and test
  - Average test set accuracy over 100 runs
  - *Self repeatability* of pathologists: 75%-84%.

Deep features	**# clusters or leaf nodes**	**Error rate**
—	—	24.79%
K-means clustering	30-60	24.02%
hClustering (various)	[10,40]	**23.46%**
rpForests	30	**23.28%**

Understanding *deepTacoma* by simulations

- Using Gaussian mixtures as data generating model
- Gaussian mixtures
  - $\mathcal{G}_1$ for usual mixture data
  - $\mathcal{G}_2$ for heterogeneous data
    - Data of same label may be from different mixture component
  - $\mathcal{G}_3$ for high dimensional data
    - Covariance matrix estimated from TMA images
- Label noise
  - Created by flipping label of ϵ proportion of instances.

Details of Gaussian mixtures

- Gaussian mixture $\mathcal{G}_1 \in \mathbb{R}^{40}$

$$\frac{1}{2}\mathcal{N}(\boldsymbol{\mu}, \Sigma) + \frac{1}{2}\mathcal{N}(-\boldsymbol{\mu}, \Sigma),$$

  $\boldsymbol{\mu} = (0.3, ..., 0.3)^T$, labeled as ‘1’ and ‘2’, respectively
- Gaussian mixture $\mathcal{G}_2 \in \mathbb{R}^{40}$

$$\frac{1}{4}\mathcal{N}(\boldsymbol{\mu_1}, \Sigma) + \frac{1}{4}\mathcal{N}(\boldsymbol{\mu_2}, \Sigma) + \frac{1}{4}\mathcal{N}(-\boldsymbol{\mu_1}, \Sigma) + \frac{1}{4}\mathcal{N}(-\boldsymbol{\mu_2}, \Sigma),$$

  $\boldsymbol{\mu_1} = (0.5, ..., 0.5, 0, ..., 0)^T$ and $\boldsymbol{\mu_2} = (0, ..., 0, 0.5, ..., 0.5)^T$, labeled as ‘1’ from first 2 components and else ‘2’
- Covariance matrix

$$\Sigma_{ij} = \rho^{|i-j|}, \text{ for } \rho \in \{0.1, 0.3, 0.5\}.$$

Experiments on Gaussian mixture $\mathcal{G}_1$

ρ	ϵ	—	K-means	hClustering	rpForests
0.1	0	8.18%	7.68%	5.16%	5.82%
	0.1	9.25%	8.90%	5.52%	6.32%
	0.2	11.16%	10.71%	6.91%	8.06%
	0.3	15.28%	15.04%	11.21%	12.25%
0.3	0	11.55%	11.08%	9.26%	9.51%
	0.1	12.32%	12.16%	9.68%	9.98%
	0.2	13.77%	13.53%	11.15%	11.61%
	0.3	18.09%	17.69%	16.17%	15.58%
0.5	0	15.81%	15.73%	14.47%	14.38%
	0.1	16.73%	16.44%	15.43%	14.97%
	0.2	17.83%	17.56%	17.09%	16.43%
	0.3	22.17%	21.87%	21.98%	19.88%

Experiments on Gaussian mixture $\mathcal{G}_2$

ρ	ϵ	—	K-means	hClustering	rpForests
0.1	0	12.69%	12.45%	9.89%	10.36%
	0.1	13.64%	13.55%	10.50%	11.53%
	0.2	15.63%	15.42%	12.38%	13.40%
	0.3	20.53%	20.18%	17.37%	18.48%
0.3	0	15.69%	15.91%	14.11%	14.14%
	0.1	17.28%	16.79%	14.95%	15.22%
	0.2	18.76%	18.61%	16.67%	16.95%
	0.3	23.41%	23.03%	22.39%	21.37%
0.5	0	19.56%	20.49%	19.85%	18.07%
	0.1	20.65%	21.33%	20.50%	19.14%
	0.2	22.63%	23.02%	23.07%	21.08%
	0.3	26.35%	26.67%	26.67%	24.44%

Experiments on Gaussian mixture $\mathcal{G}_3$

ϵ	—	K-means	hClustering	rpForests
0.1	1.58%	1.48%	1.18%	1.10%
0.2	3.42%	3.24%	3.06%	2.40%
0.3	9.48%	9.12%	8.24%	7.68%
0.4	26.50%	25.90%	26.16%	25.94%

Scoring TMA images is hard

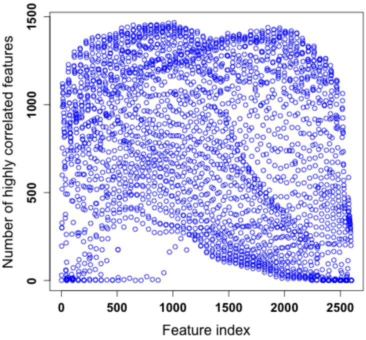


Figure: #highly correlated features ($|correlation| > 0.6$).

Summary

- *rpForests* = power of ensemble + flexibility of trees
- *rpForests* is a versatile tool
    - Efficient kNN search
        - Error rate decays exponentially with ensemble size
    - Discovering locality-aware deep features
        - Useful for heterogenous data or when labels are noisy
- *rpForests* runs on multicore or clustered computers.

The end

Thank you!

Acknowledgements

Slides based on joint work with

- Peng Gong (U. C. Berkeley and Tsinghua U)
- Honggang Wang (UMass Dartmouth)
- Timothy W. Randolph (Fred Hutchinson)
- Jian Zou (WPI)
- Zhenpeng Li (Dali U)
- Jin Wang (UMass Dartmouth)
- Yingjie Wang (UMass Dartmouth)

For more information

1. D. Yan, Y. Wang, J. Wang, H. Wang and Z. Li. K-nearest neighbor search by random projection forests. *IEEE Big Data 2018*, `arXiv:1812.11689`
2. D. Yan, T. W. Randolph, J. Zou and P. Gong. Incorporating deep features in the analysis of TMA images. *Statistics and Its Interface* (to appear), 2019. `arXiv:1812.00887`

`http://www.math.umassd.edu/~dyan/rpforests.html`