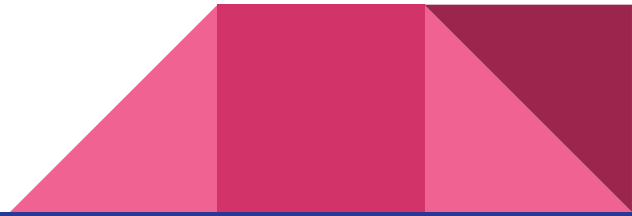
Boundary Forest

Authors: Charles Mathy, **Nate Derbinsky,** José Bento, Jonathan Rosenthal, Jonathan Yedidia

Presenters: Steven Abbott, Michael Ruberto, Jesse Steinberg, Zhizhen Zhu (Group 4)

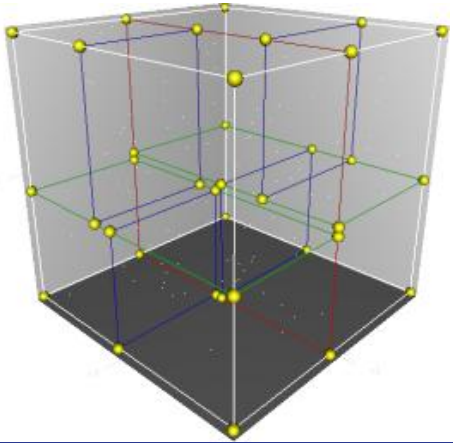
Background

- **KD Tree and Random Decision Forest**
- Friedman, J.; Bentley, J.; and Finkel, R. 1977. An algorithm for finding best matches in logarithmic expected time. ACM Trans. on Mathematical Software 3:209–226.
- **Cover Tree**
- Beygelzimer, A.; Kakade, S.; and Langford, J. 2006. Cover tree for nearest neighbor. In Proceedings of the 2006 23rd International Conference on Machine Learning.



K-Dimensional(KD) Tree

- A Kind of Binary Search Tree
- Each node represent value in k-dimension
- Each non-leaf node in K-D tree divides the space into two parts, called as half-spaces.
- Use on the searching key value in high dimension



Problem of KD Tree

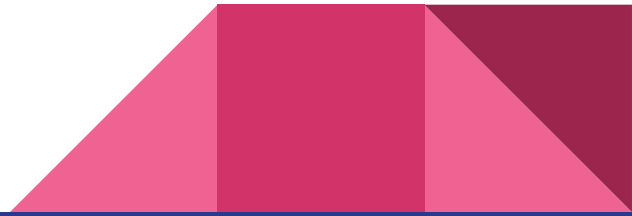
- Meant to perform cheaper computations
- In fact, kd trees typically scale no better than brute force in higher than 20 dimensions.
- Break up the features

Random decision Forest(RF)

- A type of model used for both classification and regression
- Easy to interpret and make for straightforward visualizations.
- Can handle both numerical and categorical data.
- Perform well on large datasets
- Random will reduce the variance in result

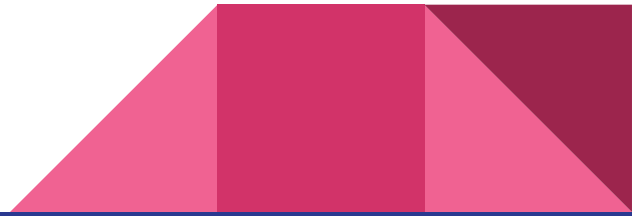
Problem of the RF Tree

- Building decision trees require algorithms capable of determining an optimal choice at each node. Which is greedy.
- Decision trees are prone to overfitting, especially when a tree is particularly deep.



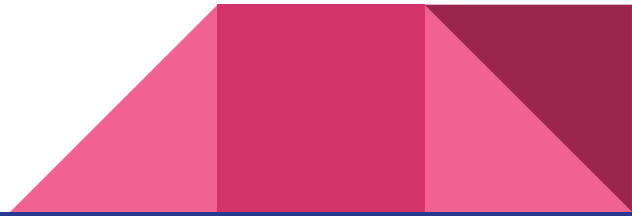
Cover Tree

- An algorithm designed to facilitate the speed-up of a nearest neighbor search
- O(c^6logN) for inserting
- where N is the amount of data and c the so-called expansion constant



Problem of Cover Tree

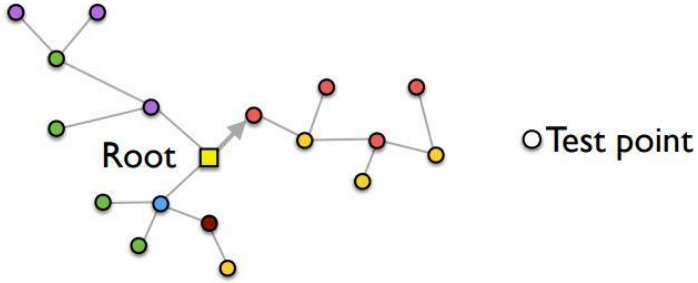
- A Tradeoff Comparing to other algorithm
- Similar Problem with K-nearest neighbor
- Does not work well with large dataset
- Does not work well with large dimension
- Sensitive to noisy data



The Boundary Forest Algorithm

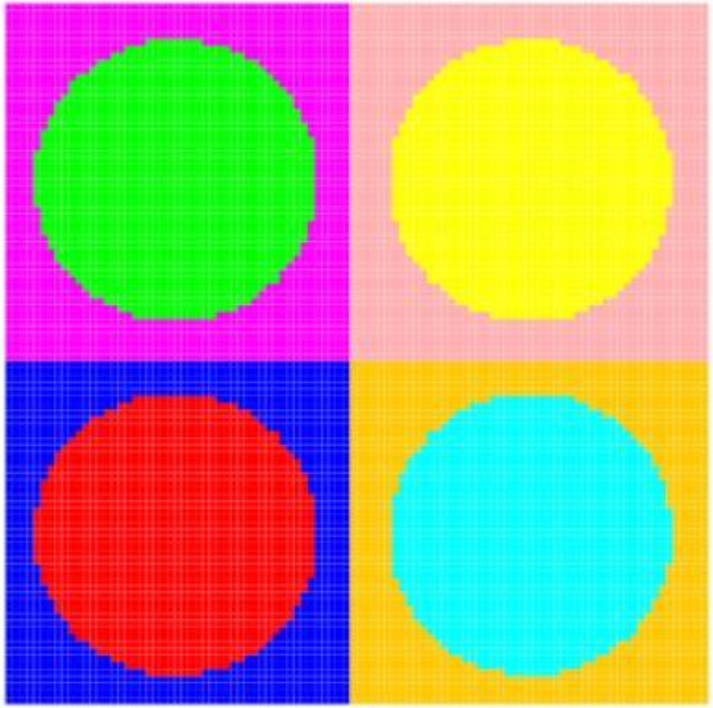
Boundary Forest - Parameters & Details

- Collection of Boundary Trees (BT's)
- Each node stores training data
- Requirements/Parameters:
  - # of trees w/in the forest (n_t)
  - Maximum # of child nodes allowed for any node in a tree (k)
  - Distance metric function *d(x, y)*
    - Any function to measure closeness between nodes and training/testing data
    - ex. Euclidean distance



Boundary Forest - Train vs. Query

- Train
  - From root, recursively compute closest node to labeled training vector
  - IF correct prediction:
    - Do nothing *
  - IF incorrect prediction:
    - Add new node & edge
- Query
  - Same traversal as training
  - Closest node to the query is the prediction

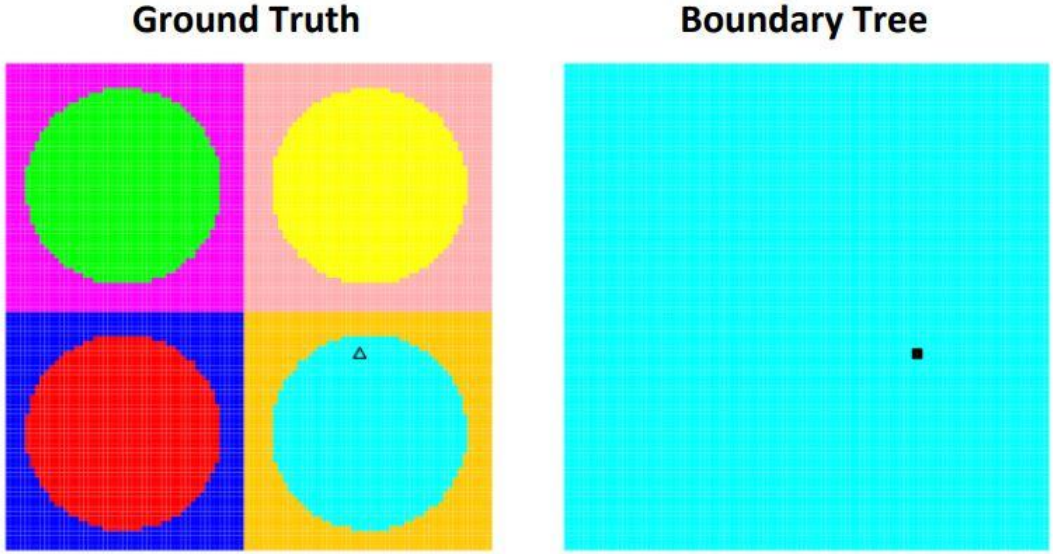


2D Classification Training Example

(6 training points)

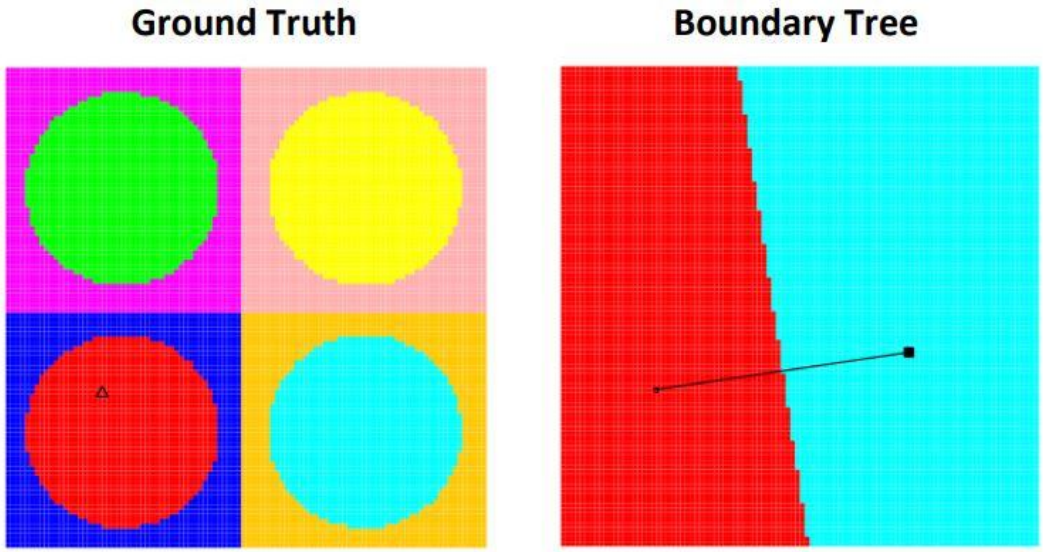
Example images from:
https://derbinsky.info/public/_custom/research/aaai2015_bf/talk.pdf

Boundary Forest - Training Example (1)



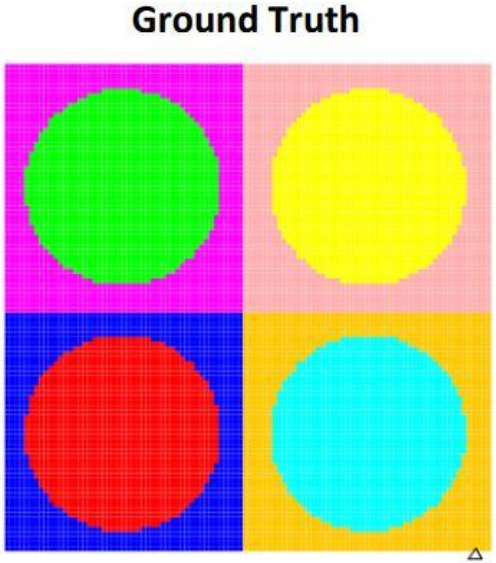
- Classes are color
- Features are (x, y)
- Point became root

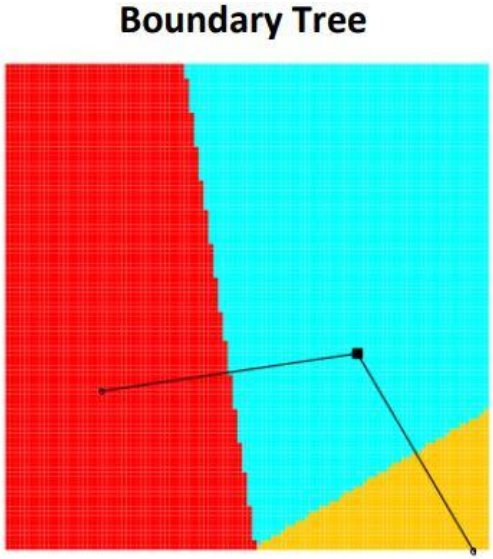
Boundary Forest - Training Example (2)



- Prediction
  - Blue
- Result
  - Add new node as child of root

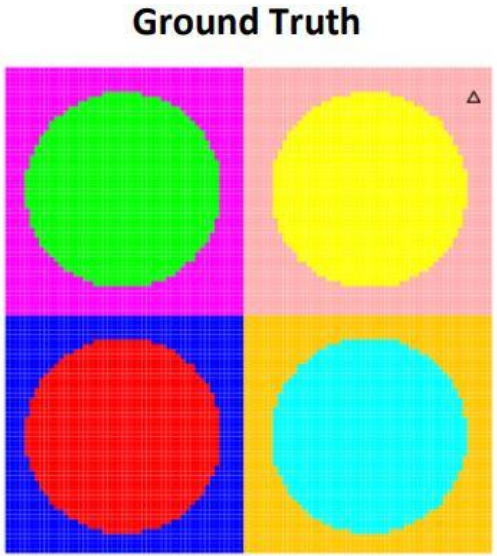
Boundary Forest - Training Example (3)

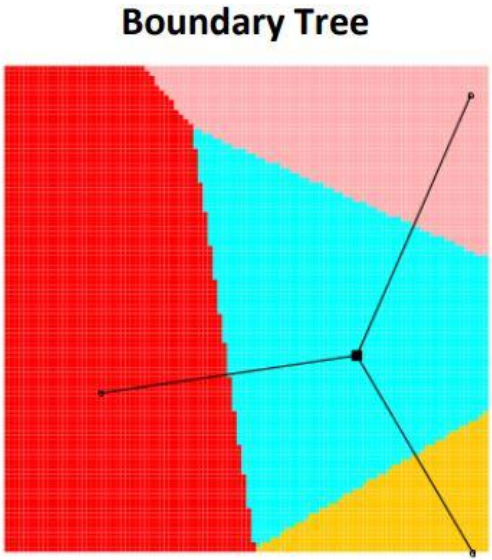




- Prediction
  - Blue
- Result
  - Add new node as child of root

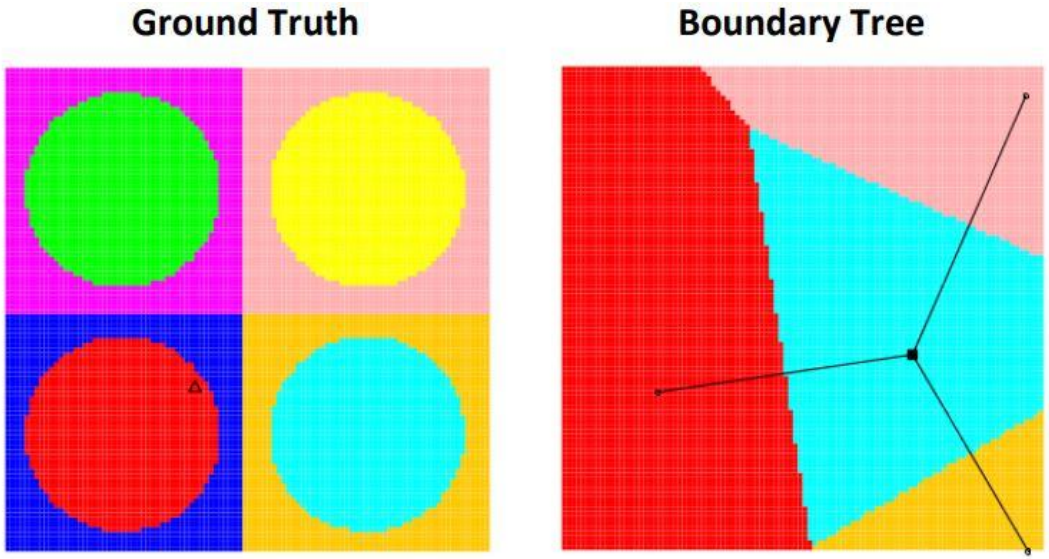
Boundary Forest - Training Example (4)





- Prediction
  - Blue
- Result
  - Add new node as child of root

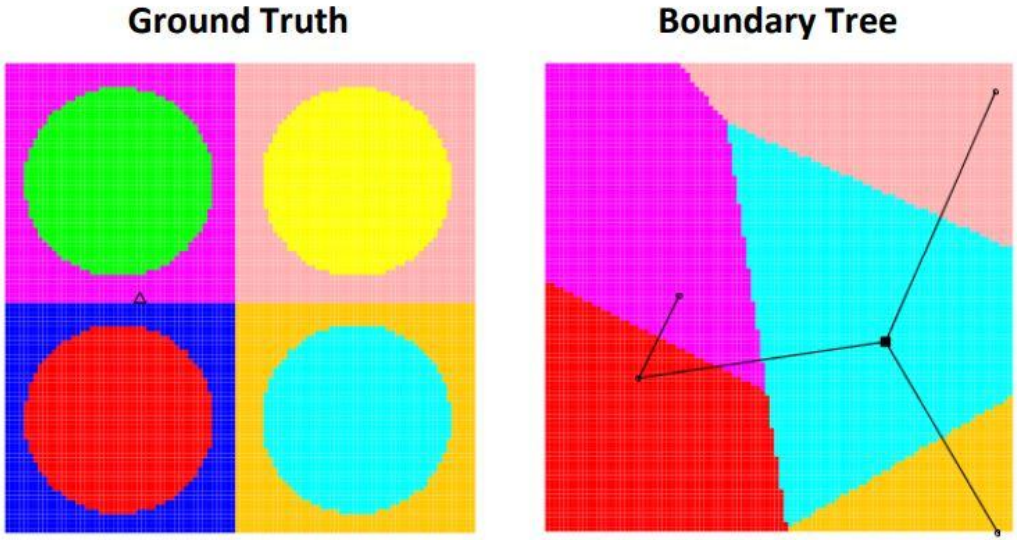
Boundary Forest - Training Example (5)



- Prediction
  - Red
- Result
  - Do not add the training point

- Important:
  - BF only keeps training points that update the model

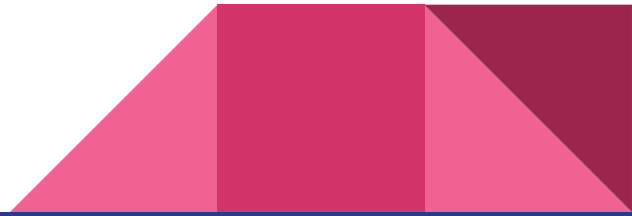
Boundary Forest - Training Example (6)



- Prediction
  - Red
- Result
  - Add new node as child of red node

Tests

- Comparisons between different classification algorithms
- Trials performed on datasets from LIBSVM Repository
- Compared Boundary Forest (BF) to k-Nearest Neighbor (k-NN), Random kd Trees (R-kd), Random Decision Forests (RF), and Cover Tree (CT) algorithms



Numerical Results – Training Time

Data	BF	BF-4	R-kd	CT	RF
dna	0.34	0.15	0.042	0.32	3.64
letter	1.16	0.80	0.12	1.37	7.5
mnist	103.9	37.1	5.67	168.4	310
pendigits	0.34	0.42	0.059	0.004	4.7
protein	35.47	13.81	0.90	44.4	191
seismic	48.59	16.30	1.86	176.1	2830

*k-NN algorithms do not have separate training times

Numerical Results – Testing Time

Data	BF	BF-4	1-NN	3-NN	R-kd	CT	RF
dna	0.34	0.15	3.75	4.23	0.050	0.25	0.025
letter	1.16	0.80	5.5	6.4	1.67	0.91	0.11
mnist	23.9	8.7	2900	3200	89.2	417.6	0.3
pendigits	0.34	0.42	2.1	2.4	0.75	0.022	0.03
protein	35.47	13.8	380	404	11.5	51.4	625
seismic	16.20	5.2	433	485	65.7	172.5	1.32

Numerical Results – Training and Testing Times

	Training					Testing						
Data	BF	BF-4	R-kd	CT	RF	BF	BF-4	1-NN	3-NN	R-kd	CT	RF
dna	0.34	0.15	0.042	0.32	3.64	0.34	0.15	3.75	4.23	0.050	0.25	0.025
letter	1.16	0.80	0.12	1.37	7.5	1.16	0.80	5.5	6.4	1.67	0.91	0.11
mnist	103.9	37.1	5.67	168.4	310	23.9	8.7	2900	3200	89.2	417.6	0.3
pendigits	0.34	0.42	0.059	0.004	4.7	0.34	0.42	2.1	2.4	0.75	0.022	0.03
protein	35.47	13.81	0.90	44.4	191	35.47	13.8	380	404	11.5	51.4	625
seismic	48.59	16.30	1.86	176.1	2830	16.20	5.2	433	485	65.7	172.5	1.32

*Highlight denotes algorithms which were outperformed by BF

Numerical Results - Error Rate

Data	BF	1-NN	3-NN	RF	R-kd	CT
dna	14.3	25.0	23.9	5.7	22.5	25.55
letter	5.4	5.5	5.4	7.6	5.5	5.6
mnist	2.24	3.08	2.8	3.2	3.08	2.99
pendigits	2.62	2.26	2.2	5.2	2.26	2.8
protein	44.2	52.7	50.7	32.8	53.6	52.0
seismic	40.6	34.6	30.7	23.7	30.8	38.9

Numerical Results - Error Rate (Online vs. Offline)

Data	BF	OBF
dna	14.3	13.1
letter	5.4	5.5
mnist	2.24	2.6
pendigits	2.62	2.60
protein	44.2	41.7
seismic	40.6	39.6

Conclusion

- A new online learning algorithm
- Nlog(N) training, log(N) querying
- Similar or superior performance to other Nearest Neighbor algorithms