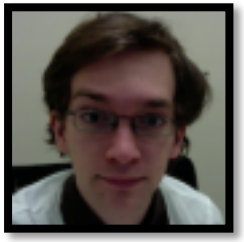
The Boundary Forest Algorithm for Online Supervised and Unsupervised Learning

Nate Derbinsky
Wentworth Institute of Technology

Charles Mathy
Disney Research



Jonathan Rosenthal
Disney Research

José Bento
Boston College

Jonathan Yedidia
Disney Research

The Problem

Approximate complicated functions
Approximate NN -> Classification, Regression

Requirements
- Incremental
- Fast to train & query
- Scale well given a large number of examples/dimensions

Potential Application Areas
- Real-time learning (e.g. vision, robotic control)
- Scalable optimization/simulation

Boundary Forest

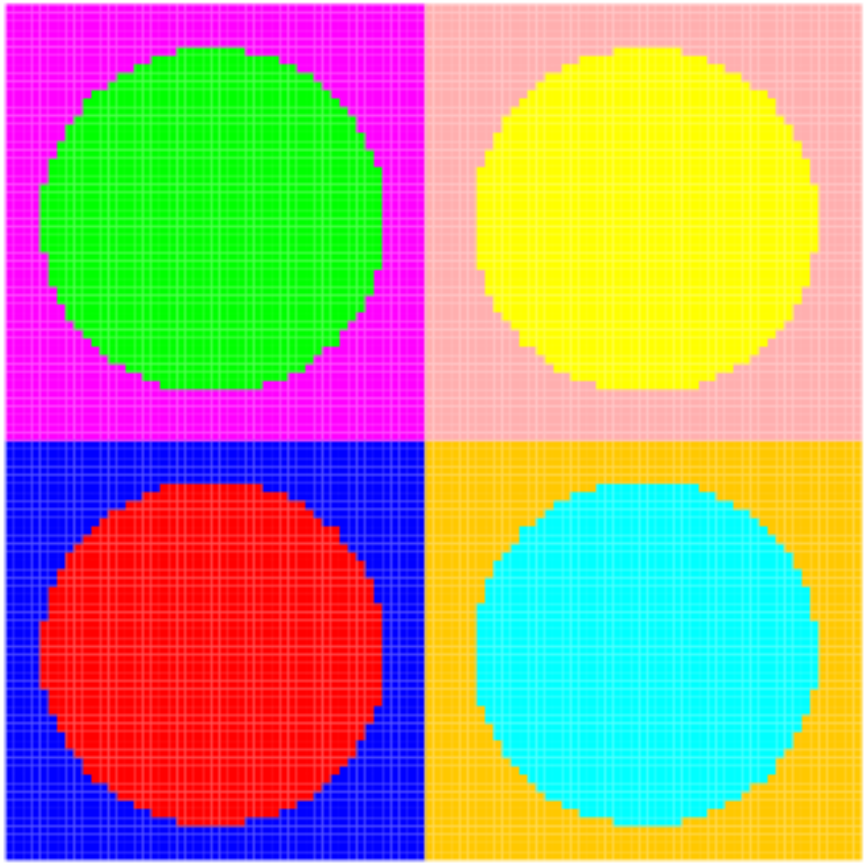
Online algorithm that performs effectively and efficiently

- Accuracy: ~ kNN
- Time: $\mathcal{O}(Dlog(N))$ – both train & query
- Memory: $\mathcal{O}(DN)$
- Incremental, fast, flexible (non-parametric, metric-based)

Ensemble of Boundary Trees, each…

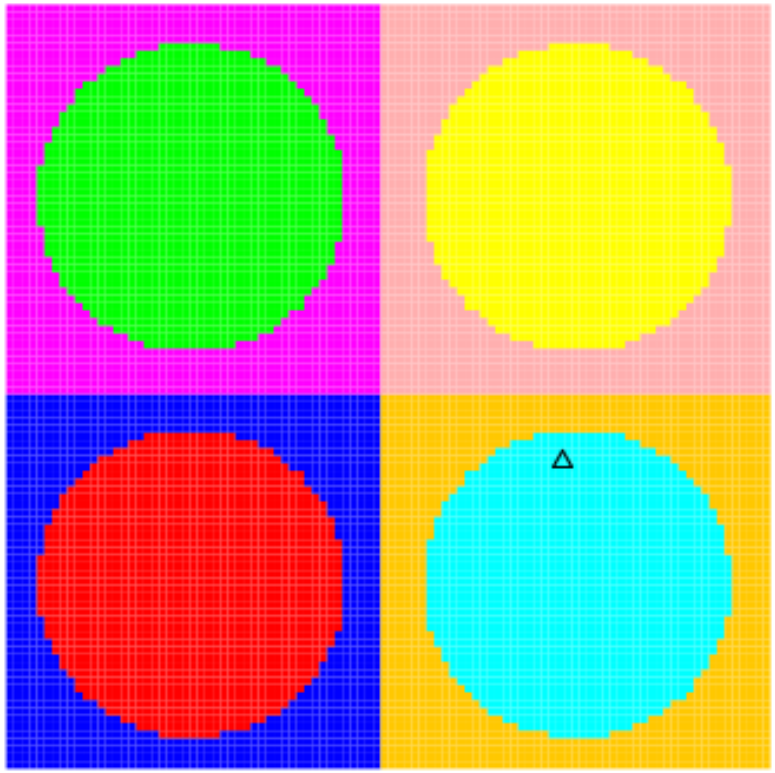
- stores a subset of examples
  - only those that inform "boundaries" (similar to incremental Condensed NN)
- incrementally builds a graphical search structure
  - queries/trains by **greedily** following/appending-to a search tree w.r.t. distance metric *d*(*x*, *y*)

A 2D Classification Example

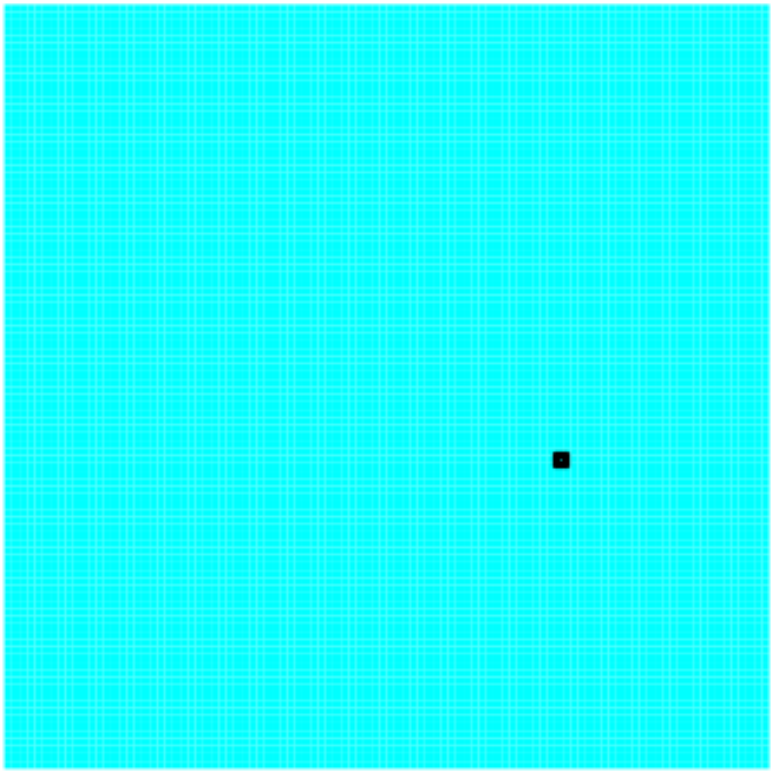


Interleaved Train/Query (1)

Ground Truth

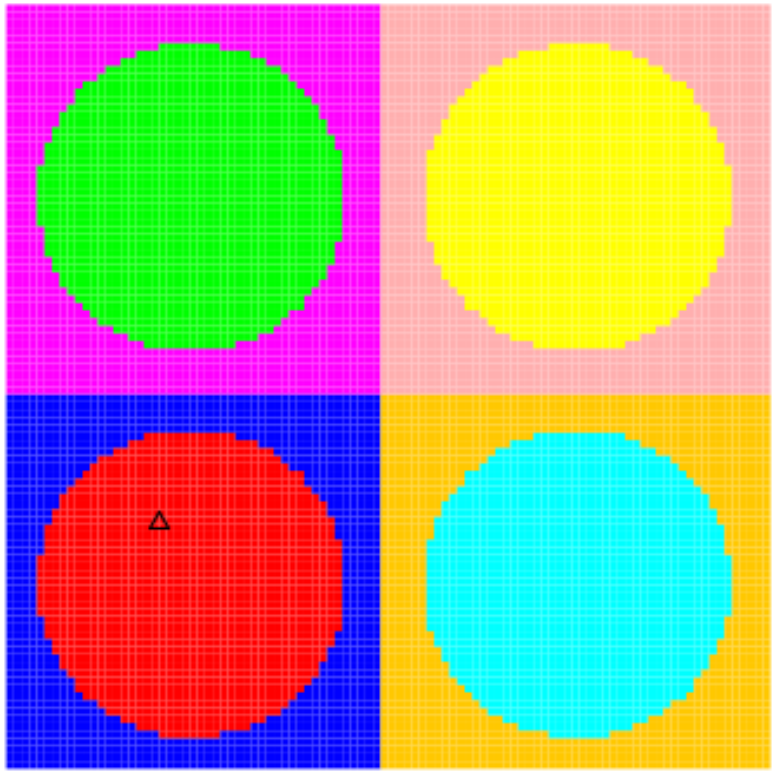


Boundary Tree

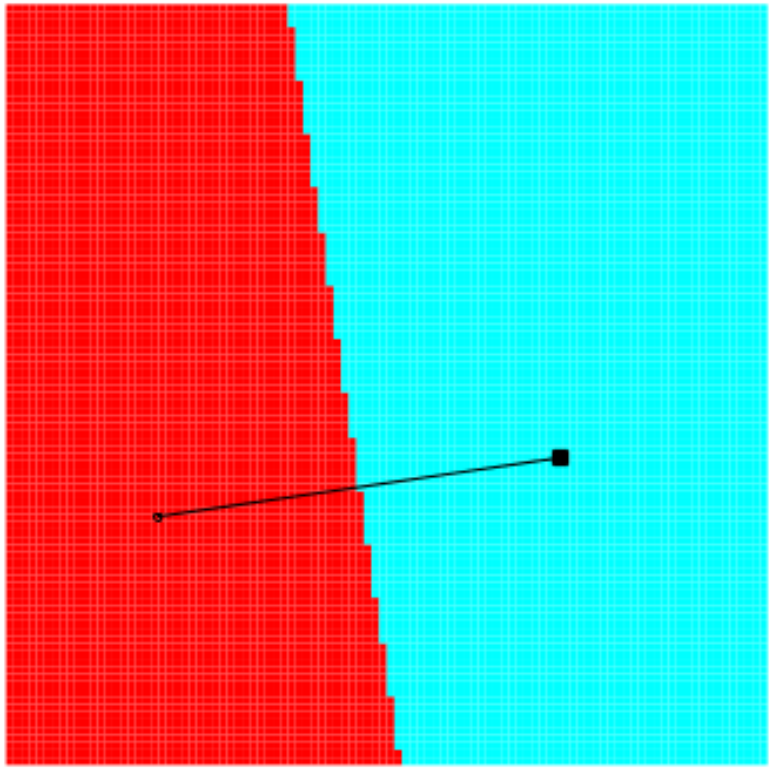


Interleaved Train/Query (2)

Ground Truth

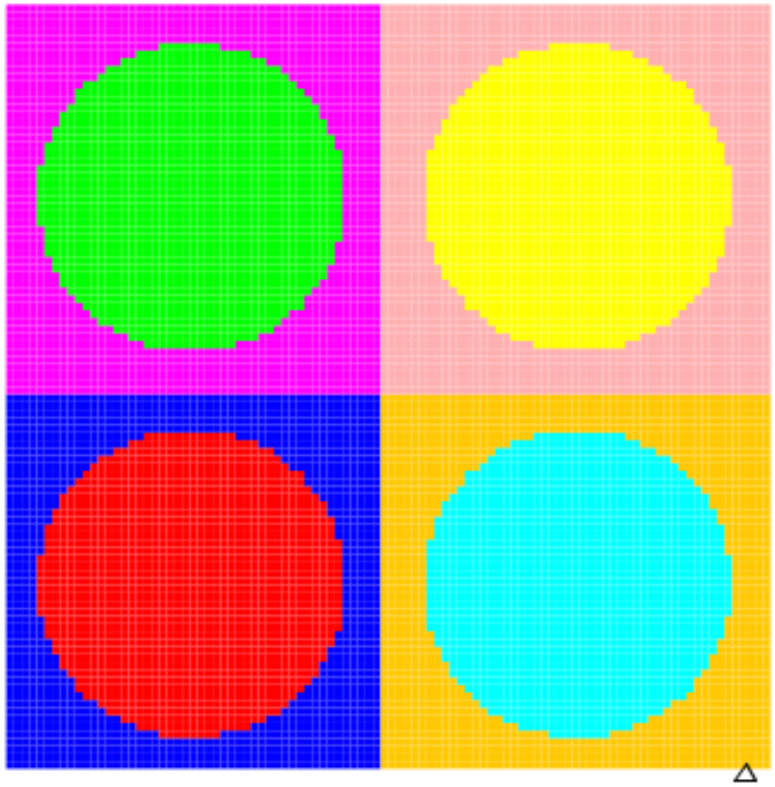


Boundary Tree

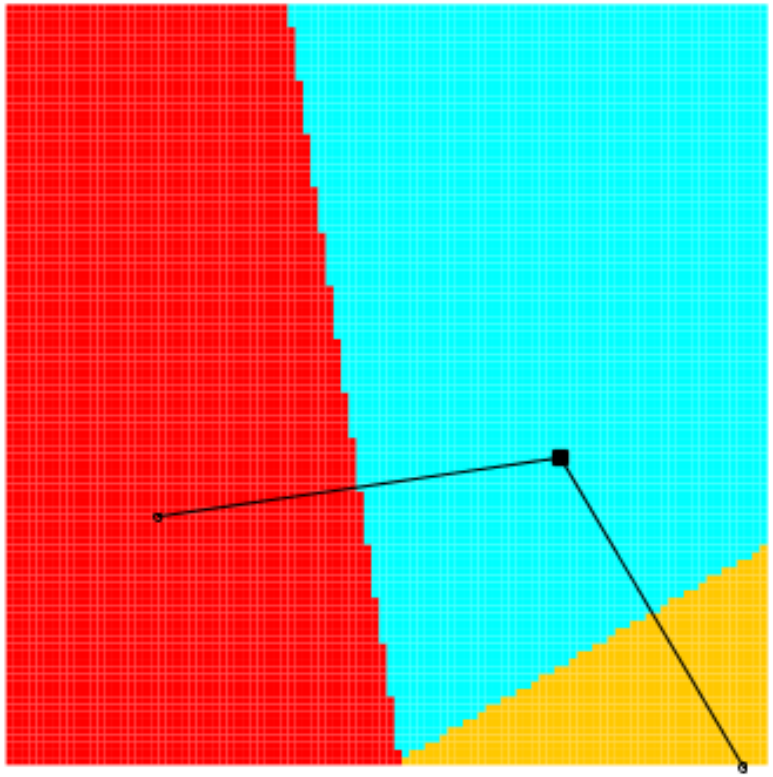


Interleaved Train/Query (3)

Ground Truth

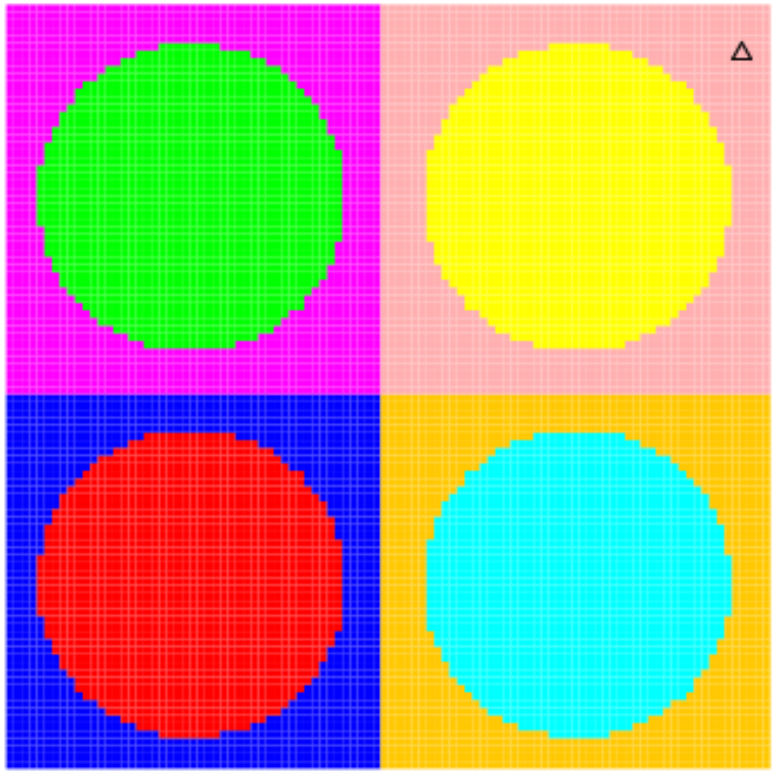


Boundary Tree

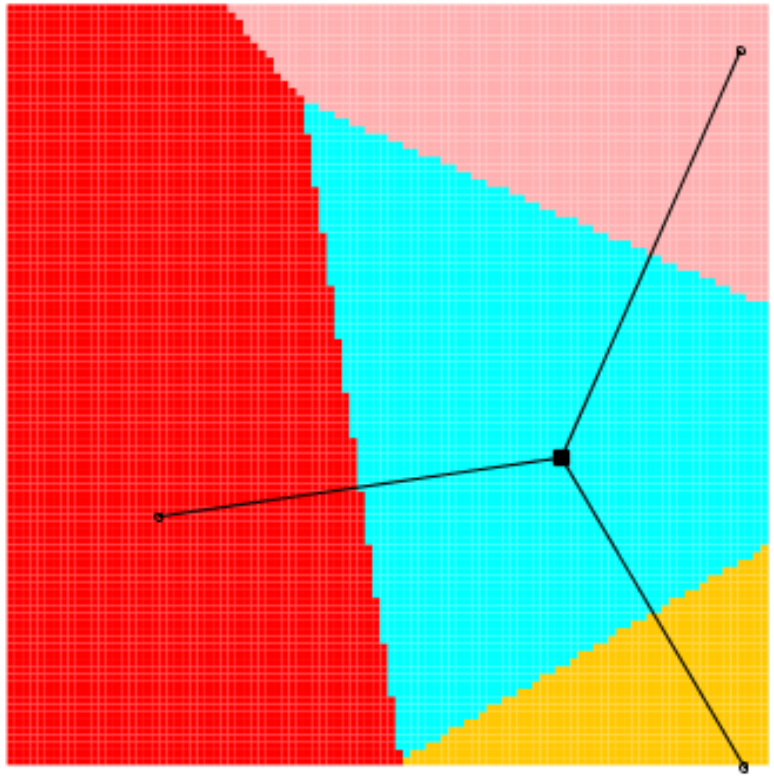


Interleaved Train/Query (4)

Ground Truth

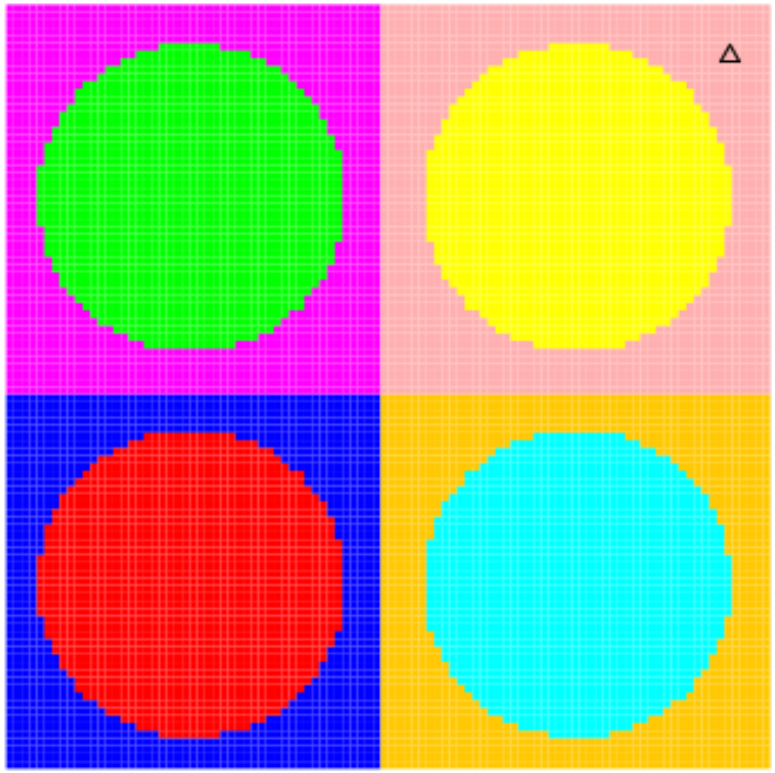


Boundary Tree

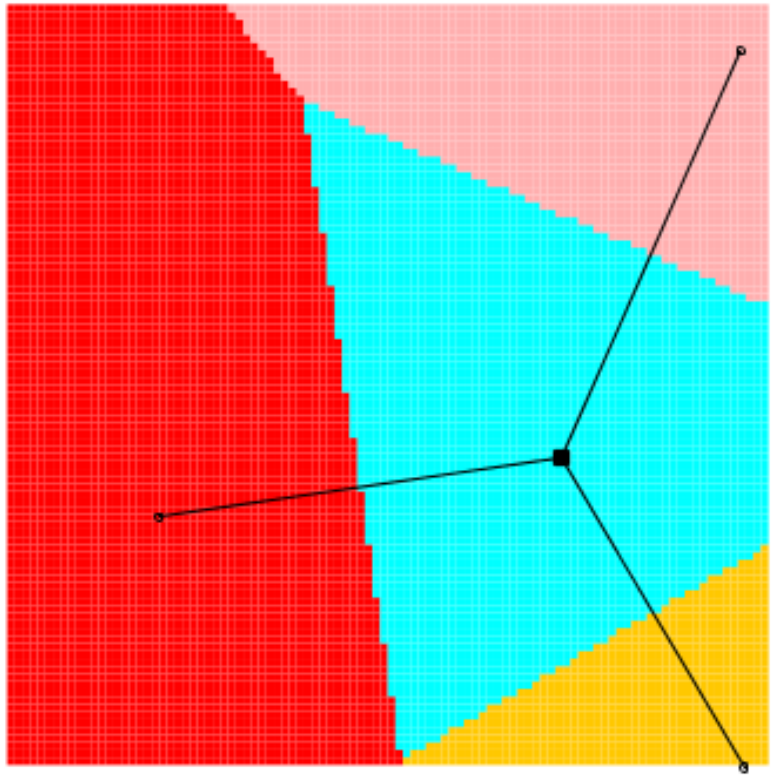


Interleaved Train/Query (5)

Ground Truth

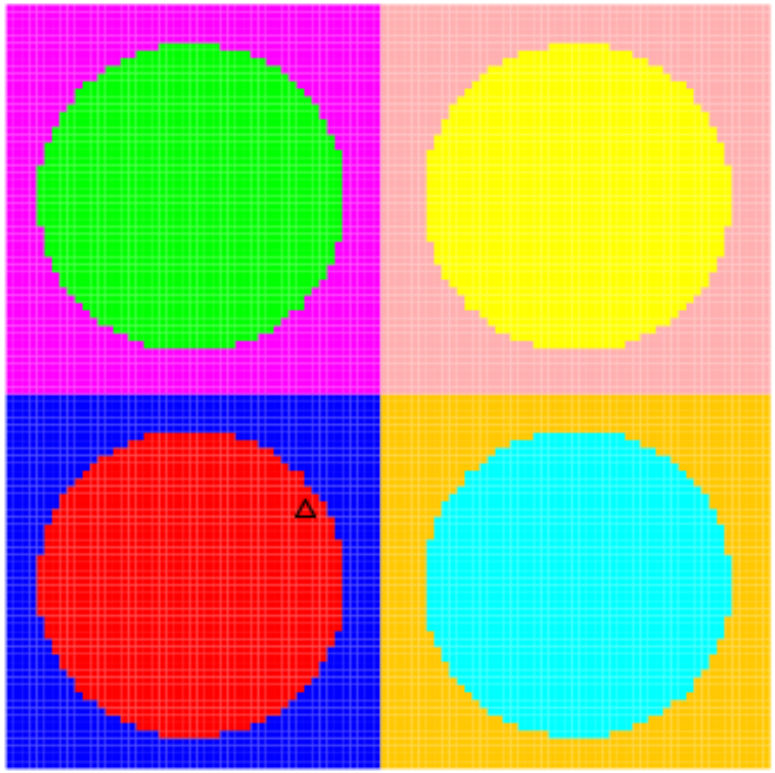


Boundary Tree

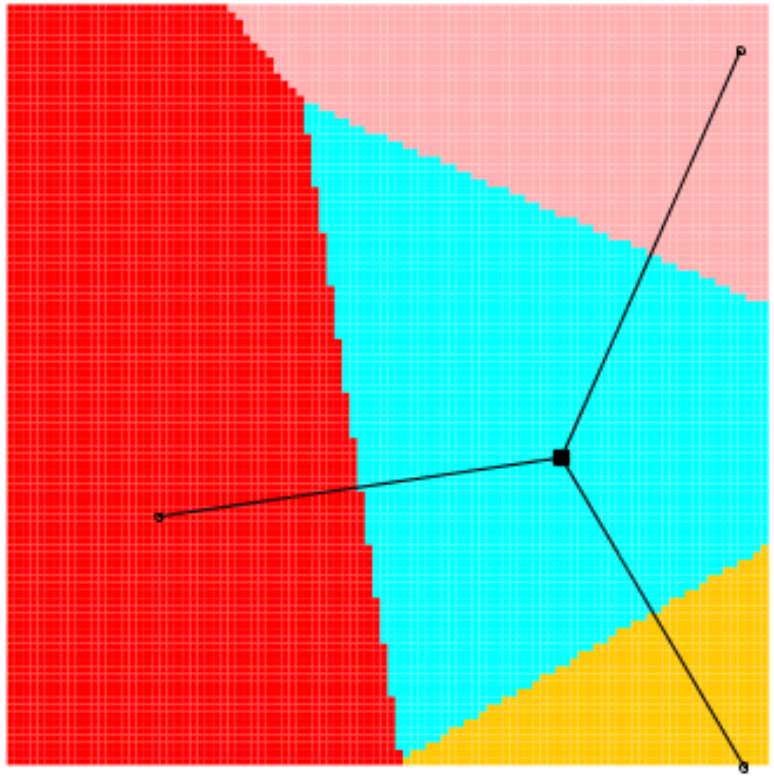


Interleaved Train/Query (6)

Ground Truth

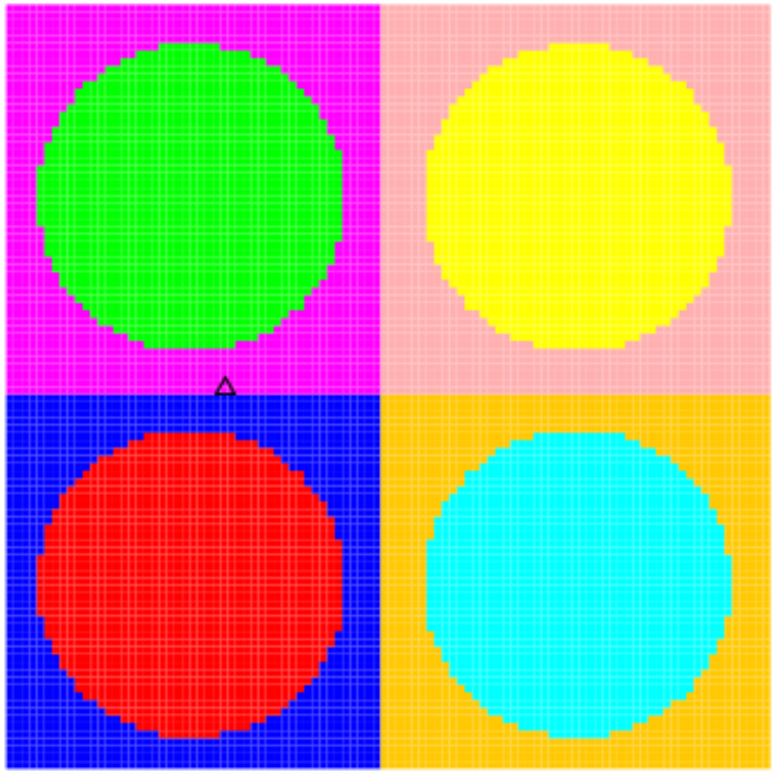


Boundary Tree

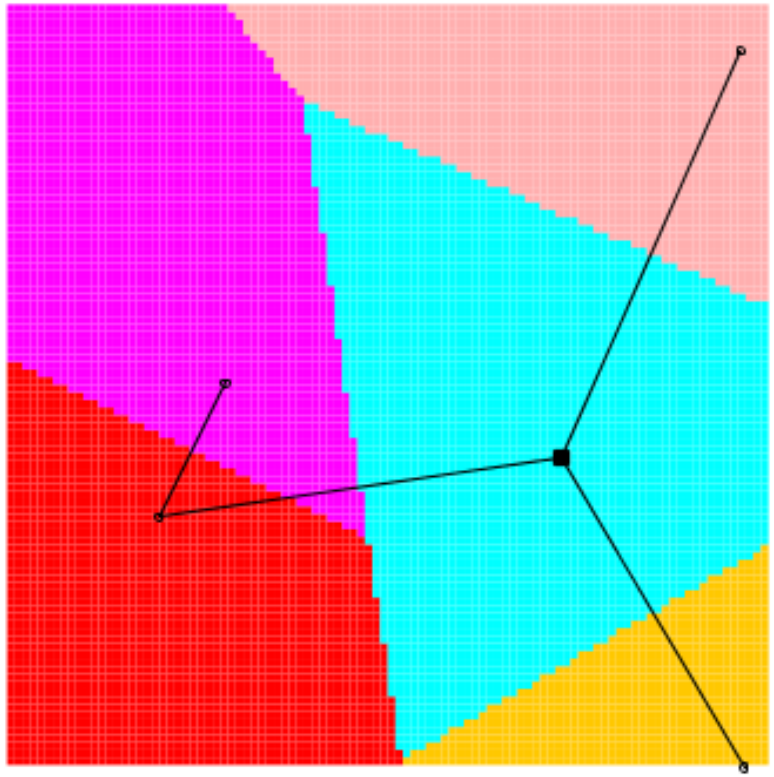


Interleaved Train/Query (7)

Ground Truth

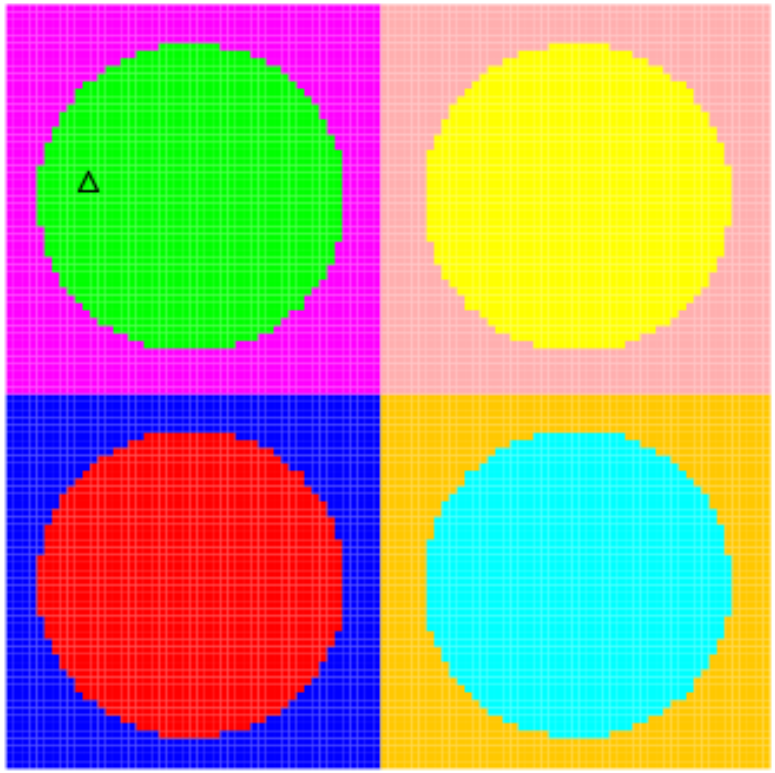


Boundary Tree

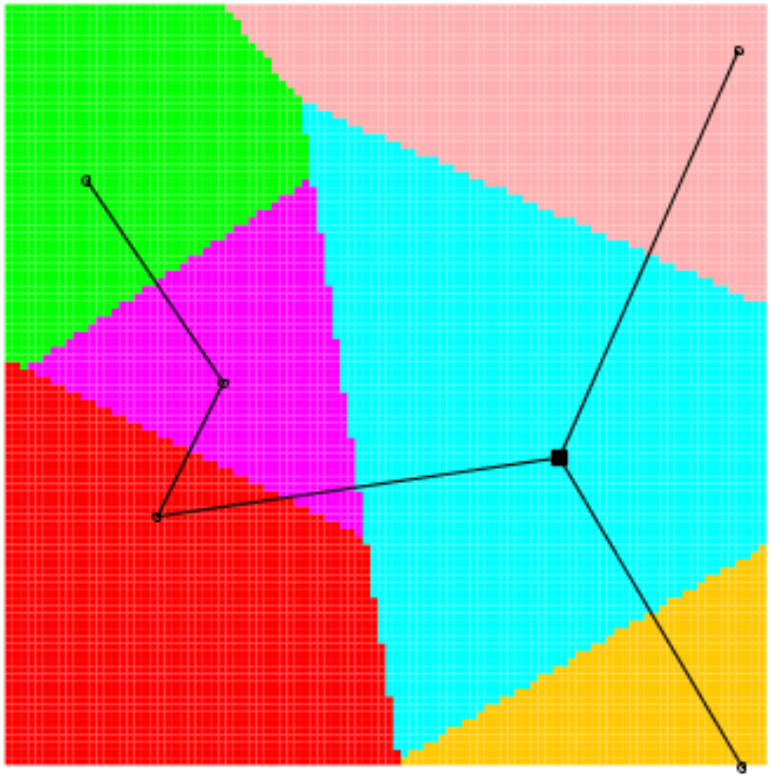


Interleaved Train/Query (8)

Ground Truth

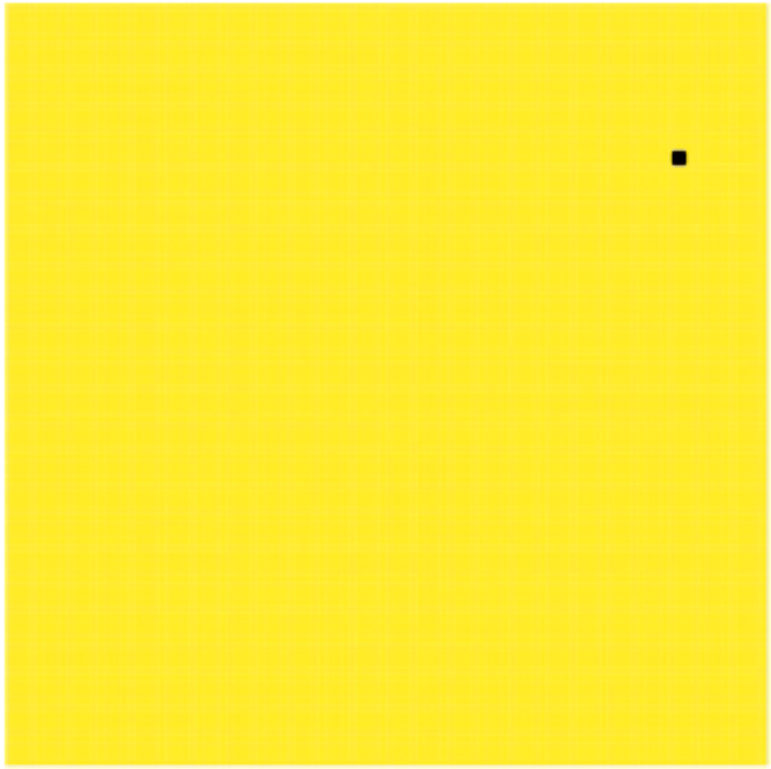


Boundary Tree

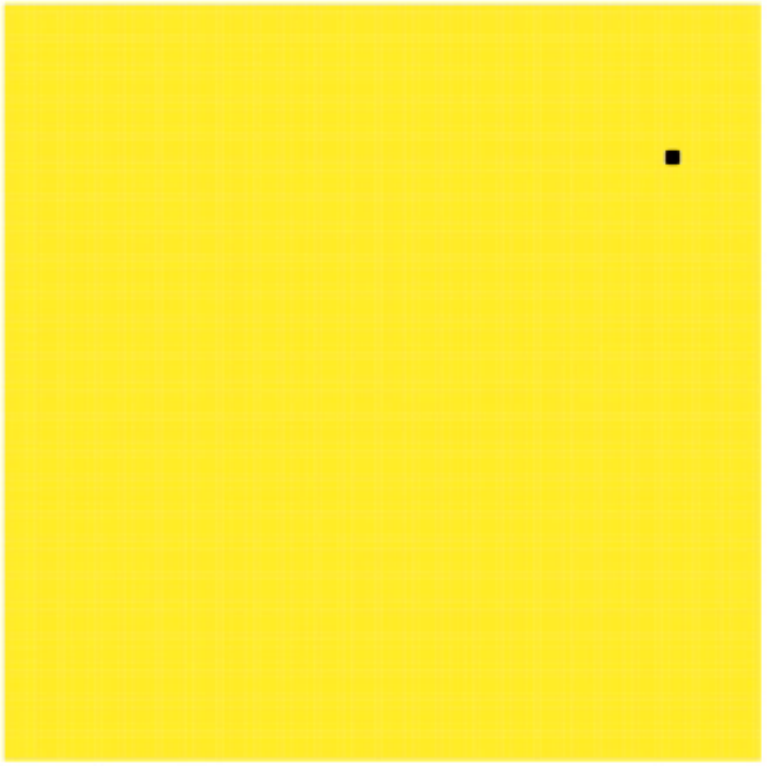


Performance & Scaling

Boundary Tree



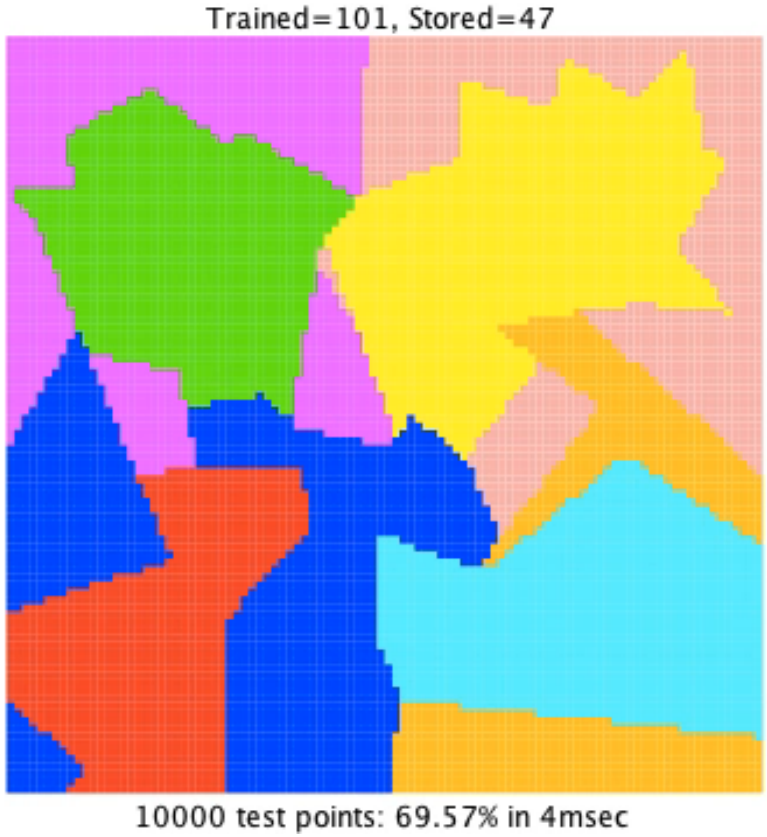
1-NN via Linear Scan



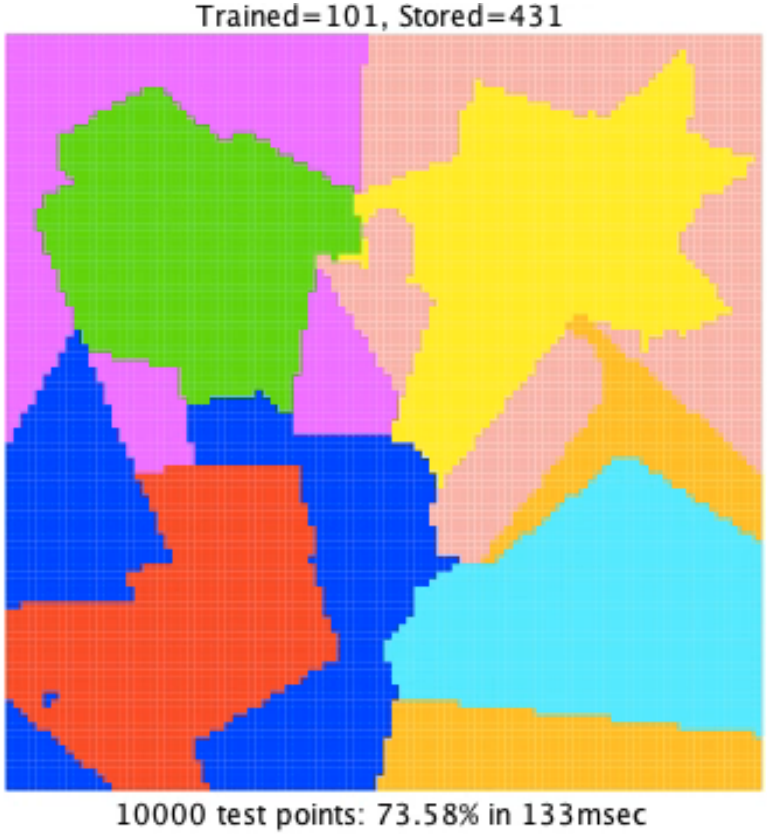
Improving Accuracy via Forests

Linear increase in memory + time

1 Tree



10 Trees



Classification Results: MNIST

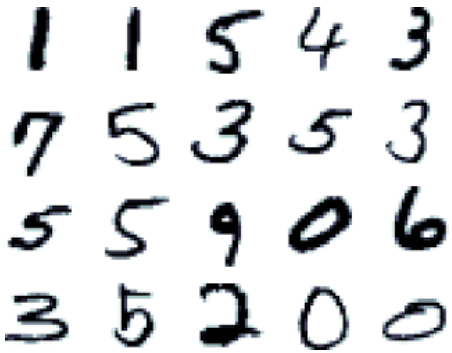
60k training, 10k testing, 784 raw pixels

Wall Clock Time (seconds)

	Training	Testing	Total
BF[1](50,50)	103.9	23.9	**127.8**
BF[4](50,50)	37.1	8.7	**45.8**
1-NN	0.0	2900.0	2900.0
3-NN	0.0	3200.0	3200.0
RF(50,50)	310.0	0.3	310.3
CT(10,1.3)	168.4	417.6	586.0

2.79X
70%

< 2 msec train
< 2.5 msec query



Error, Euclidean Distance

**HoG: 1.1%*

BF(1,50)	1-CNN	RF(50,50)	1-NN	CT(10,1.3)	3-NN	BF(50, 50)
12.15%	6.70%	3.20%	3.08%	2.99%	2.80%	**2.24%**

Additional Results in the Paper

- Power law NN accuracy
- Logarithmic NN speed (10^6 examples)
  - Theoretical analysis
- In-depth scaling comparison with CT
- Additional data sets, parameters

Algorithm Sketch
Required Parameters

- n_t = number of trees
- k = maximum outdegree
  - Typically leads to eventual logarithmic scaling
- $d(x, y)$ = distance metric
  - Need not be true metric
  - No assumptions made about properties

Algorithm Sketch
Boundary Tree

Query(y)

- v = root
- loop
  - $cand$ = children(v)
  - if |children(v)| < k
    - $cand = cand \cup v$
  - $v_{min} = argmin_{w < cand}\ d(w, y)$
  - if $v_{min} = v$: break;
  - $v = v_{min}$

Result

- NN: v_{min}
- Classification: class(v_{min})
- Regression: value(v_{min})

Train(y)

- n = Query(y)
- if ShouldAdd(n, y)
  - Connect(n, y)

ShouldAdd

- NN: True
- Classification: Diff. Class
- Regression: Diff. by ε

Algorithm Sketch
Boundary Forest

Query(y)

- for t_i : trees
  - result[i] = t_i.Test(y)

Train(y)

- for t_i : trees
  - t_i.Train(y)

Result

- NN: smallest d
- Classification: 1/d vote
- Regression: 1/d average

Initialization

- Root(t_i) = example[i]
- r = remaining (n_t-1)
  - t_i.Train(Rand(r, i))

Take-Home Points

- The Boundary Forest is a new algorithm for online classification, regression, and retrieval
- Fast both at training and testing time
- Memory and computation time grow slowly
- Easy to implement and compares favorably to other Approximate Nearest Neighbor online algorithms

Thank You :)

Questions?

Nate Derbinsky
Assistant Professor
Wentworth Institute of Technology
http://derbinsky.info

PostDoc Opportunity
José Bento, Boston College
jose.bento@bc.edu