The Boundary Forest Algorithm for Fast Online Learning of High-Dimensional Data

Nate Derbinsky

with: Charles Mathy, Jonathan Rosenthal, José Bento, Jonathan S. Yedidia

Disney Research

The Problem

Approximate complicated functions

Approximate NN, Classification, Regression

<u>Requirements</u>

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

Boundary Forest

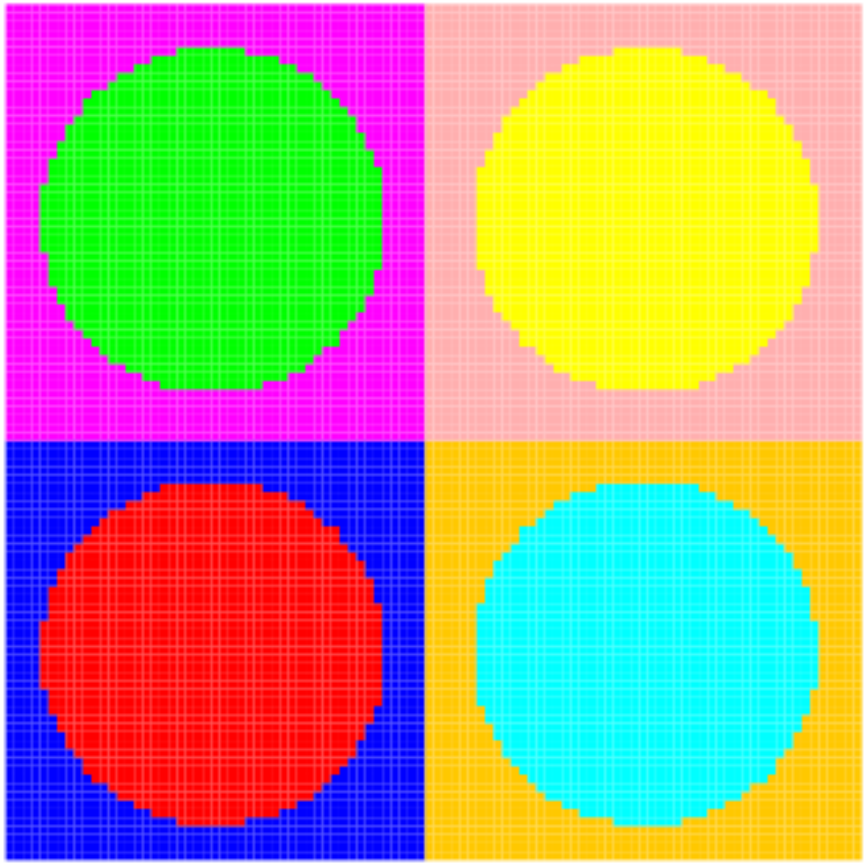
Online algorithm that performs effectively and efficiently

- Accuracy: ~kNN
- Time: *O*(logN), both train & query
- Memory: *O*(N)

Composed of Boundary Trees, each…

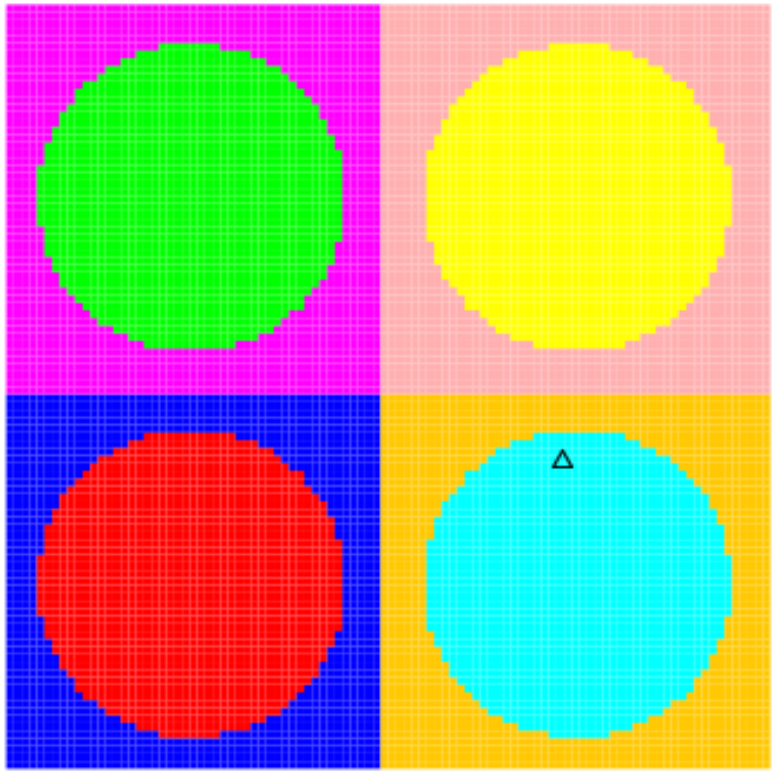
- stores a subset of examples (i.e. instance-based/non-parametric)
  - only those that inform "boundaries" (similar to incremental Condensed NN)
- incrementally builds a graphical search structure
  - queries/trains by **greedily** following/appending-to a search tree w.r.t. distance metric *d*(*x*,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

A 2D Classification Example

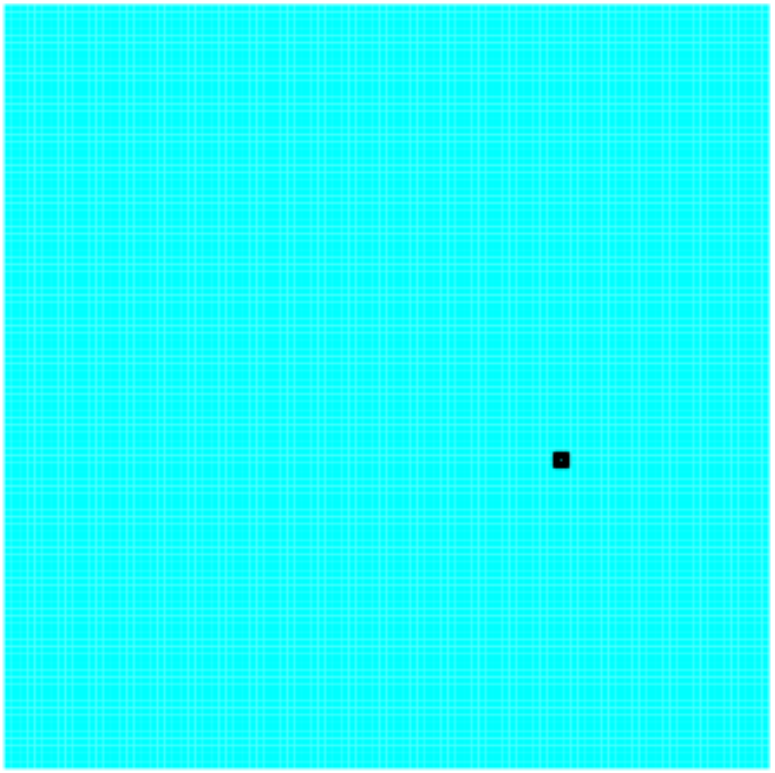


Interleaved Train/Query (1)

Ground Truth

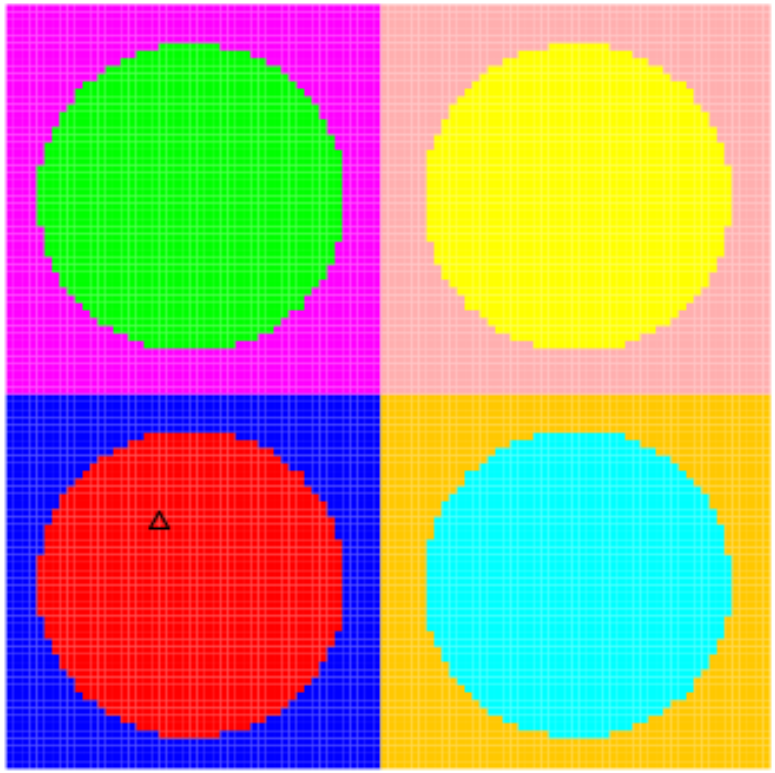


Boundary Tree

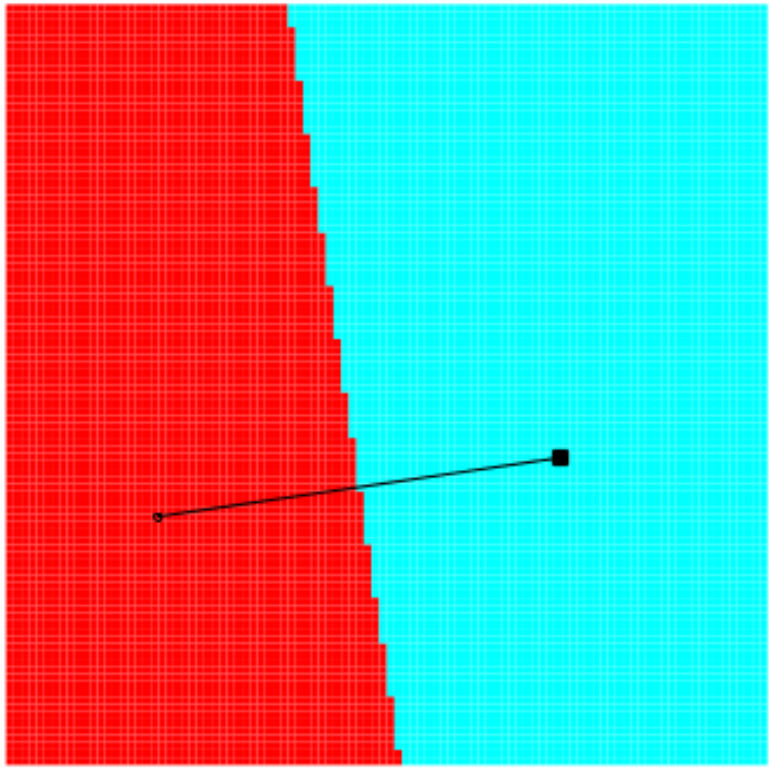


Interleaved Train/Query (2)

Ground Truth

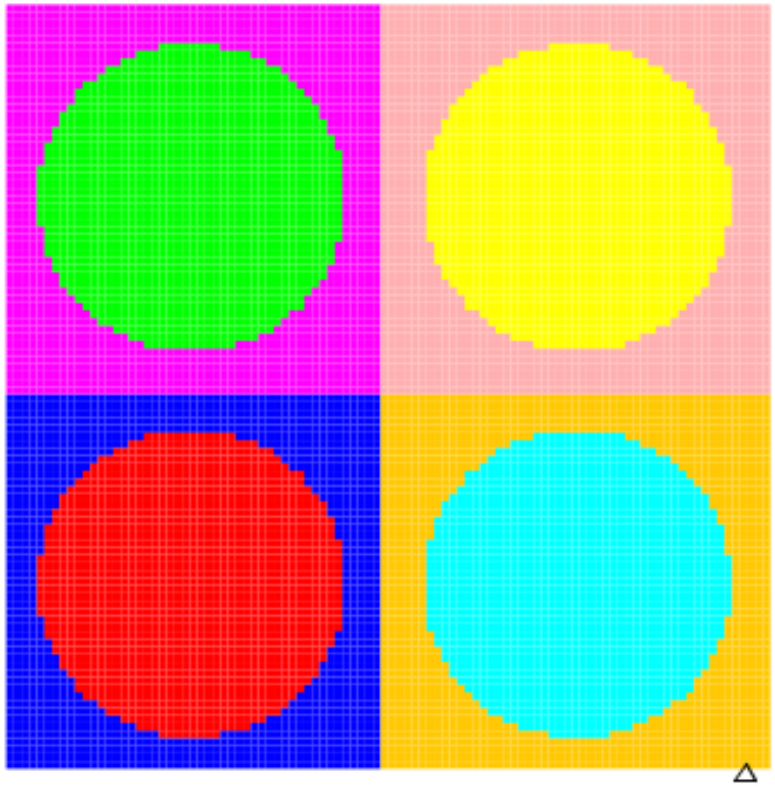


Boundary Tree

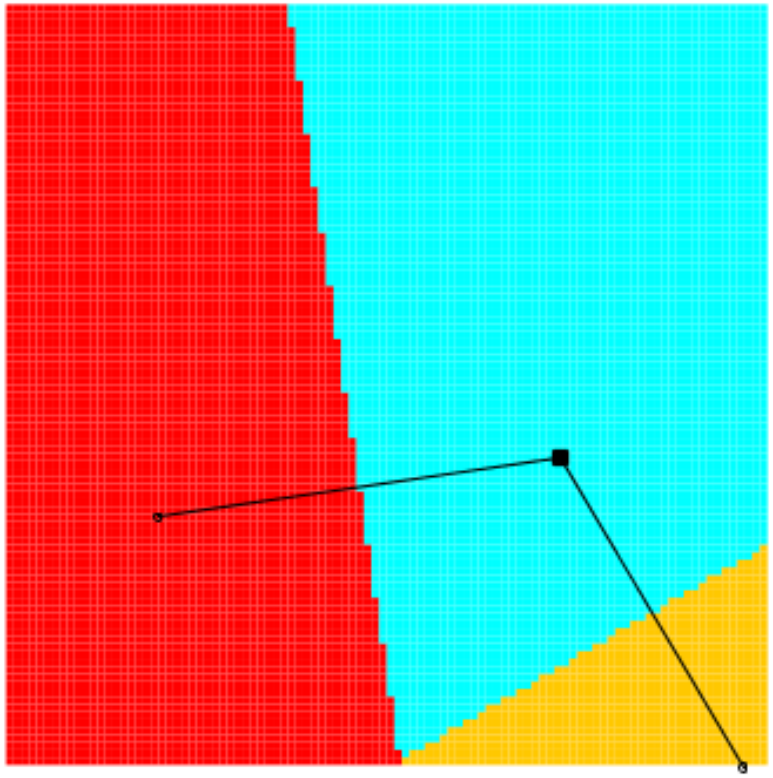


Interleaved Train/Query (3)

Ground Truth

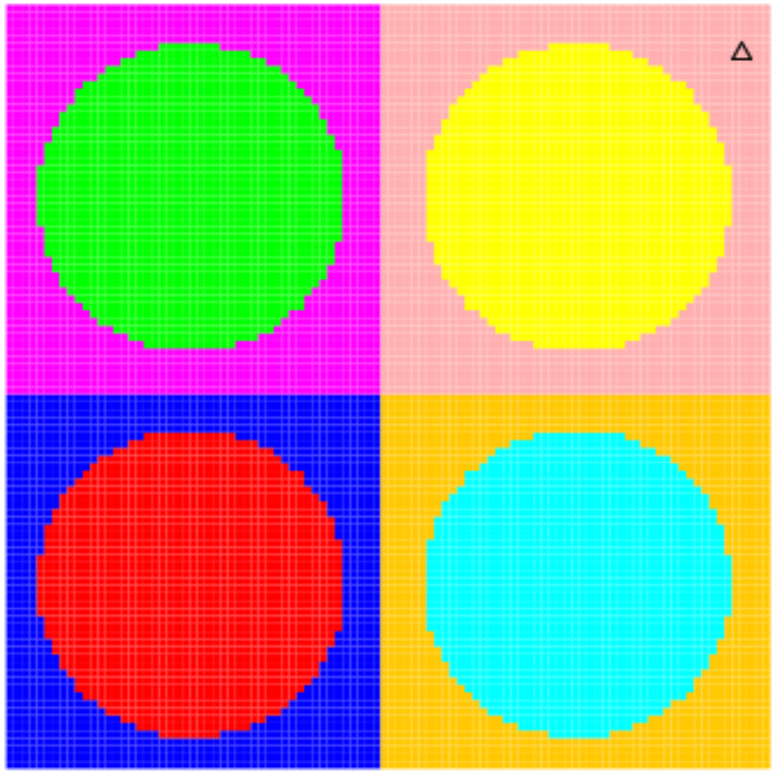


Boundary Tree

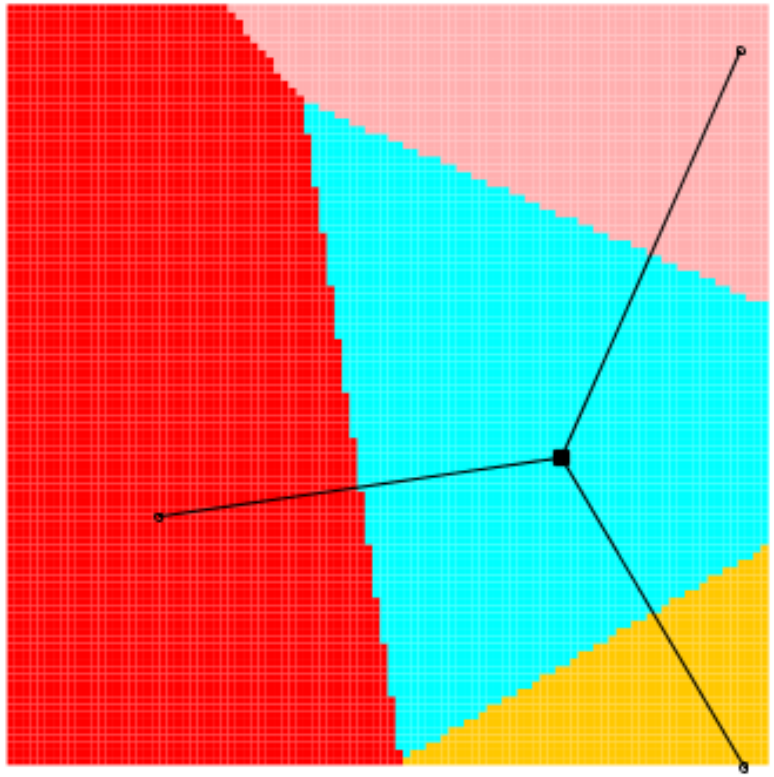


Interleaved Train/Query (4)

Ground Truth

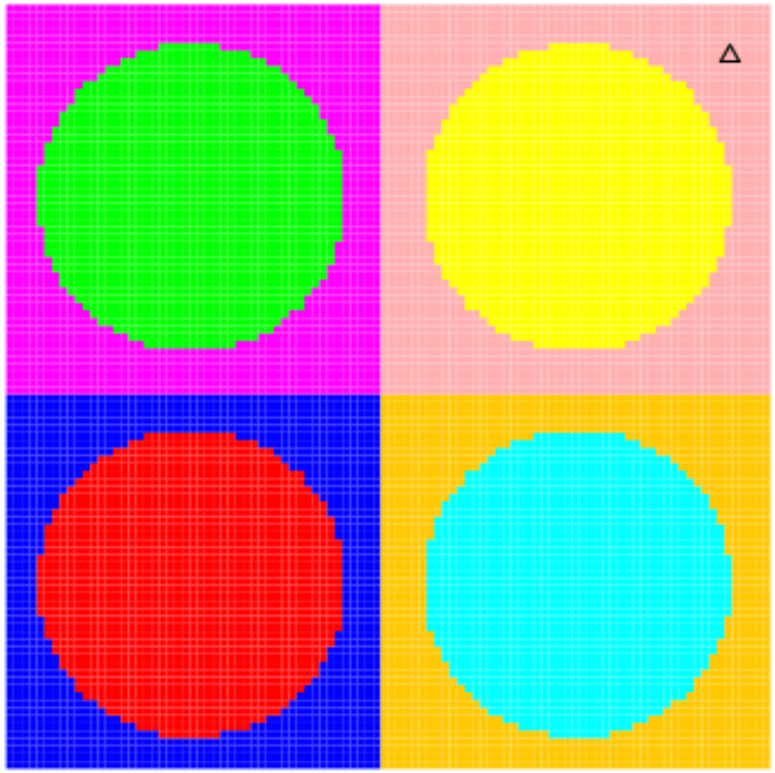


Boundary Tree

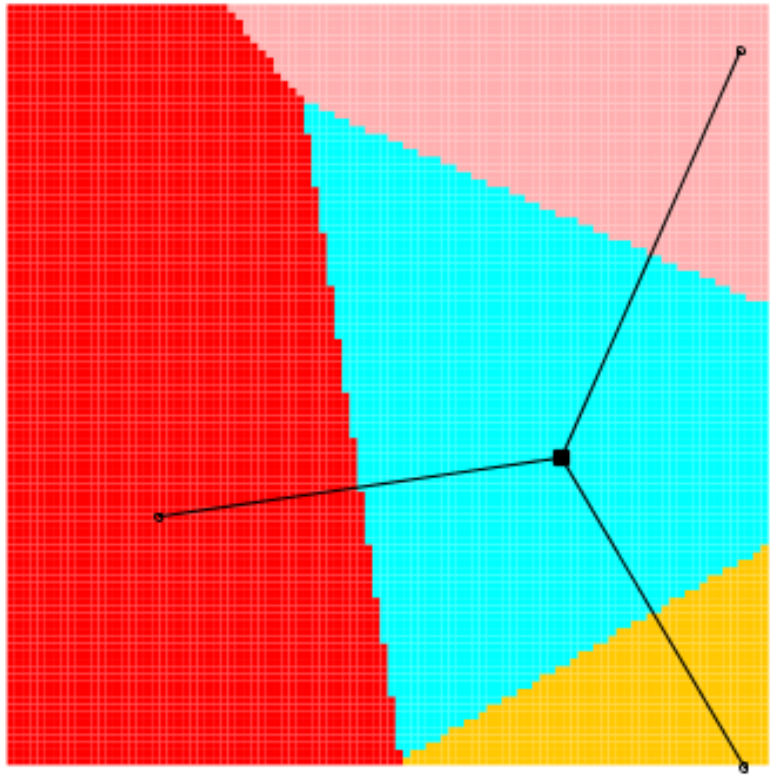


Interleaved Train/Query (5)

Ground Truth

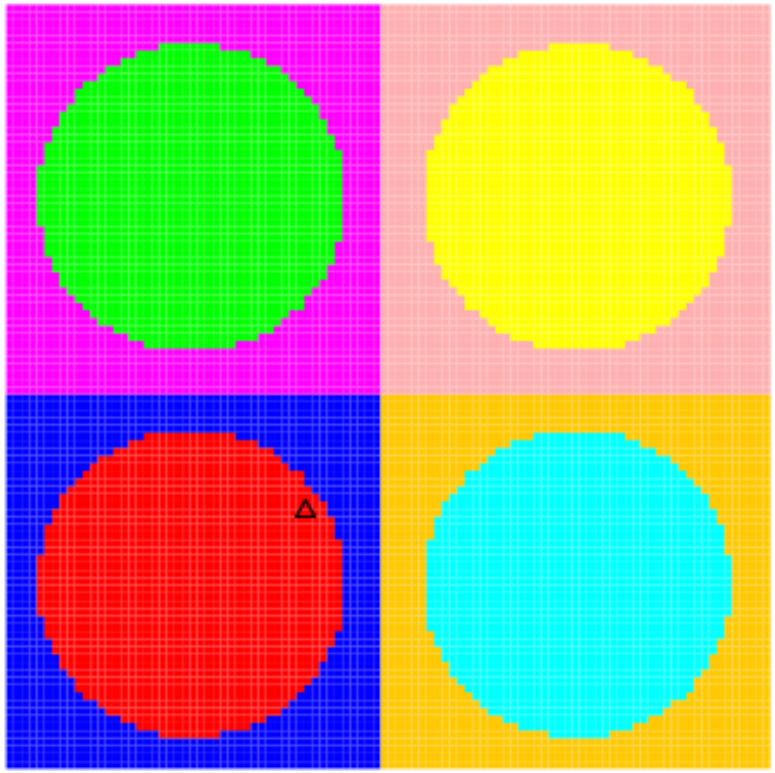


Boundary Tree

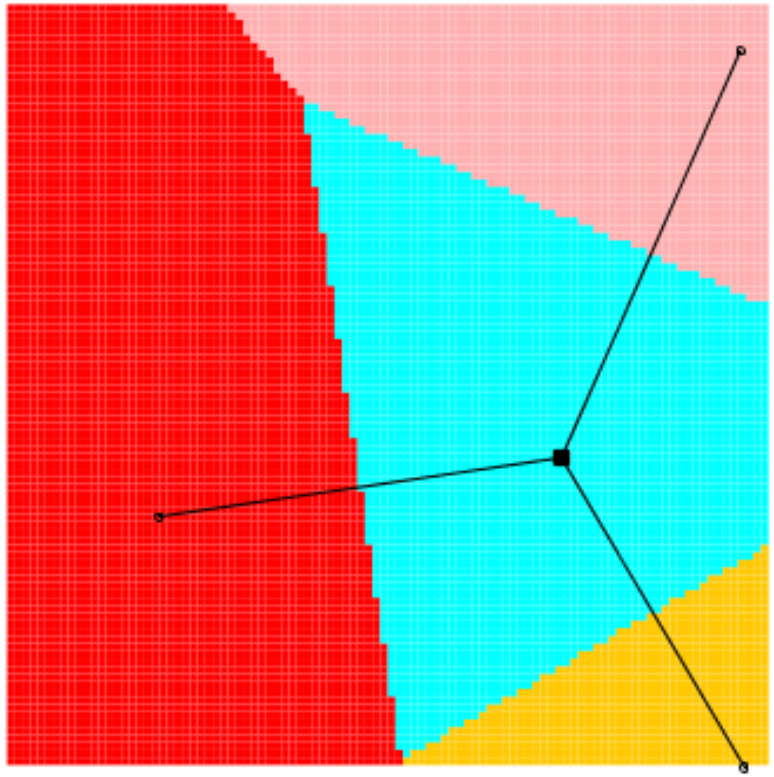


Interleaved Train/Query (6)

Ground Truth

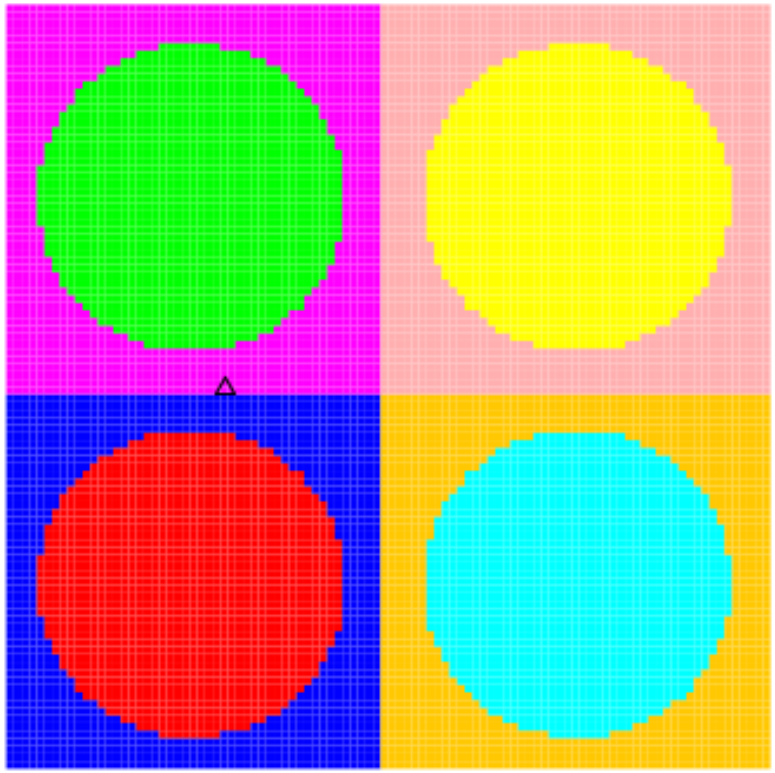


Boundary Tree

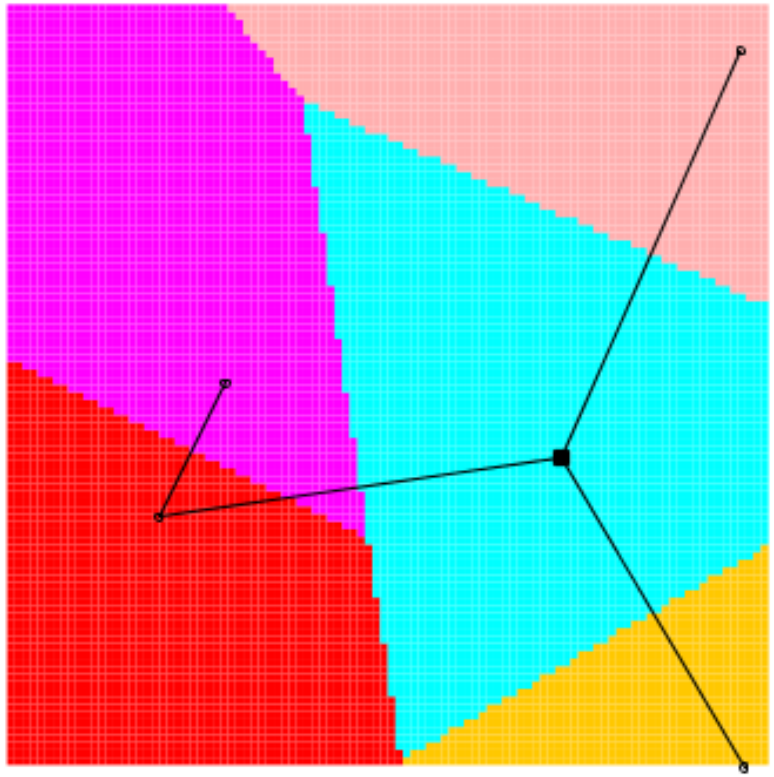


Interleaved Train/Query (7)

Ground Truth

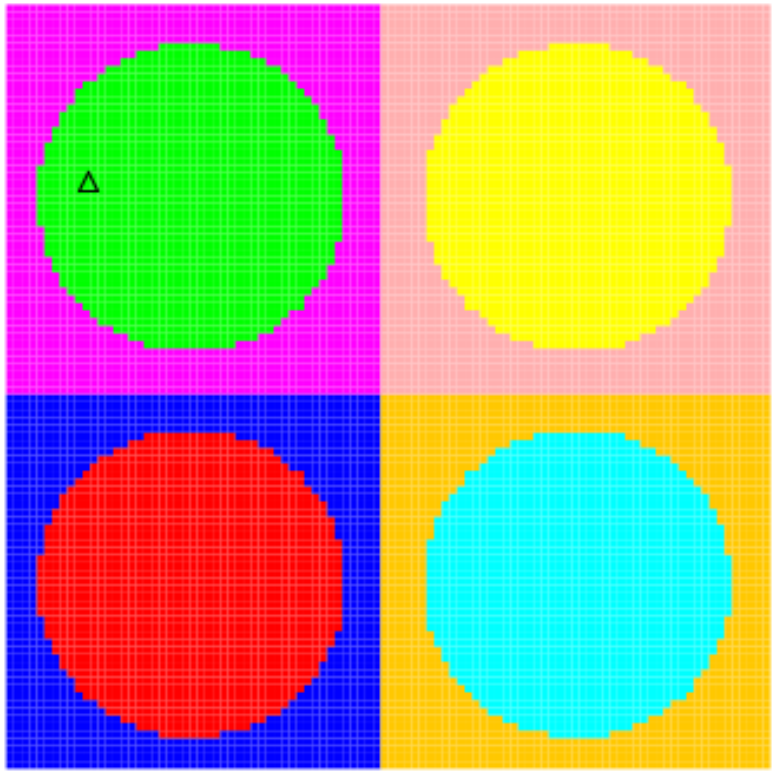


Boundary Tree

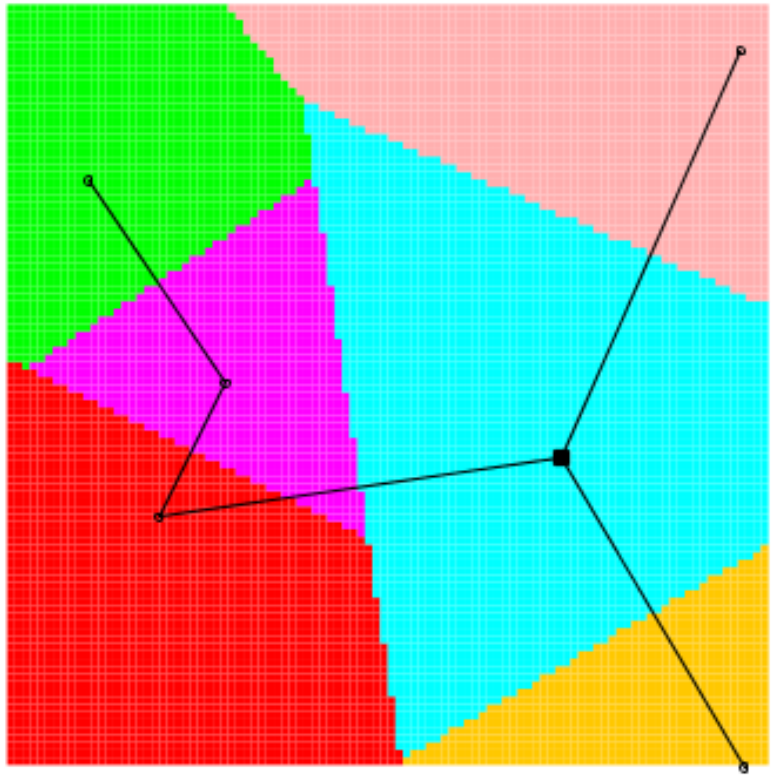


Interleaved Train/Query (8)

Ground Truth

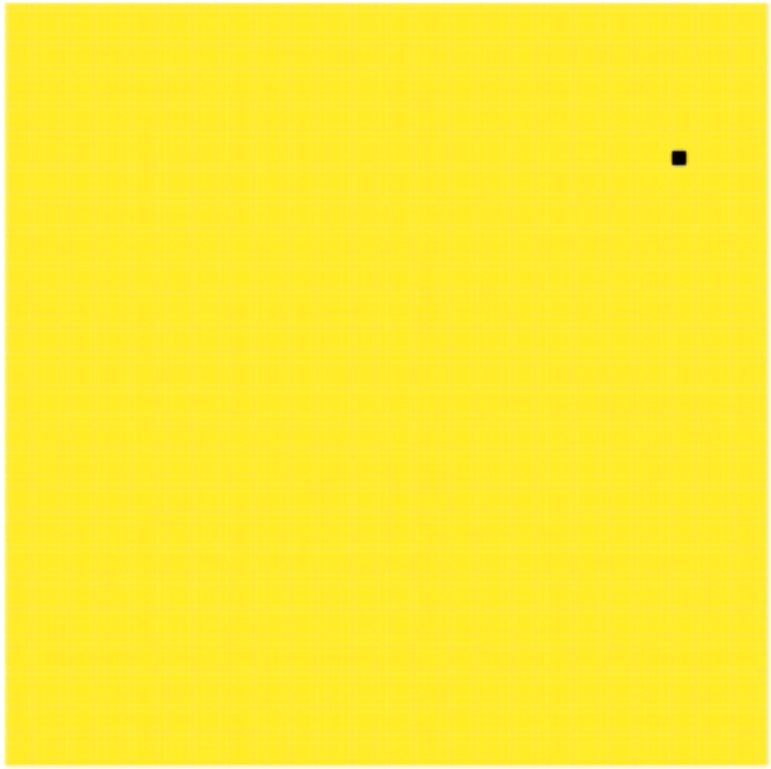


Boundary Tree

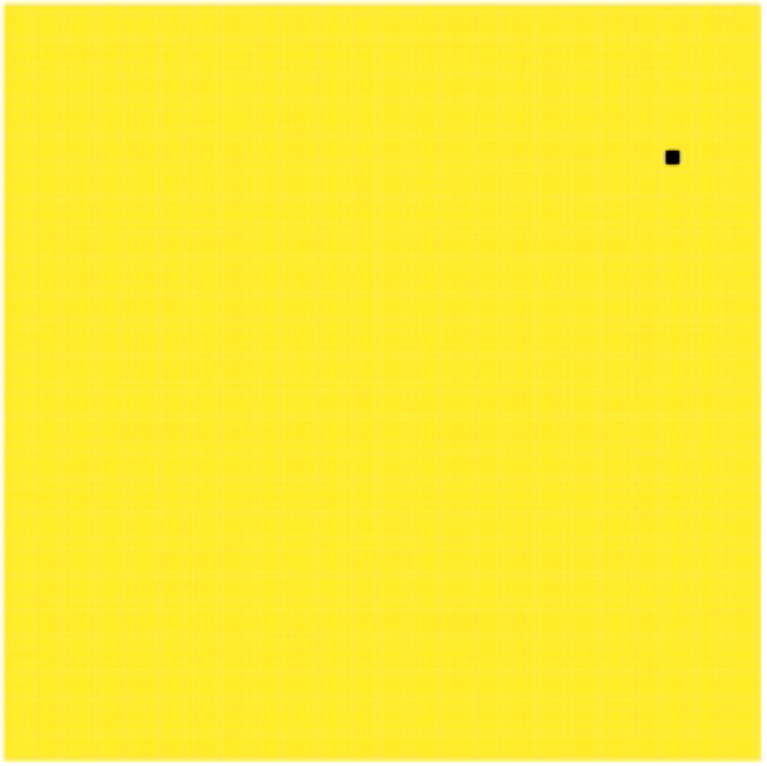


Performance & Scaling

Boundary Tree



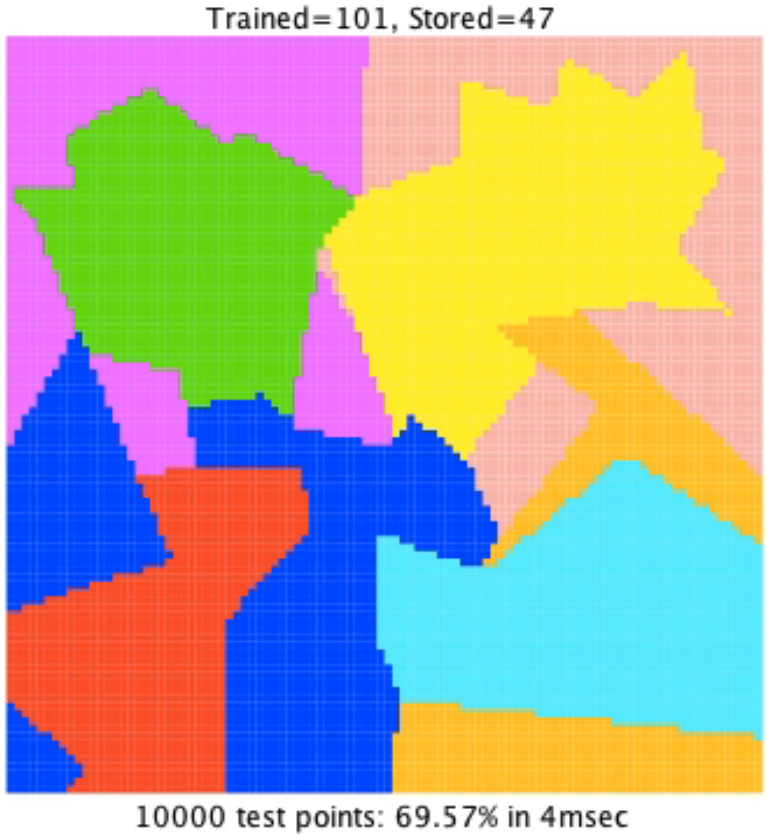
1-NN via Linear Scan



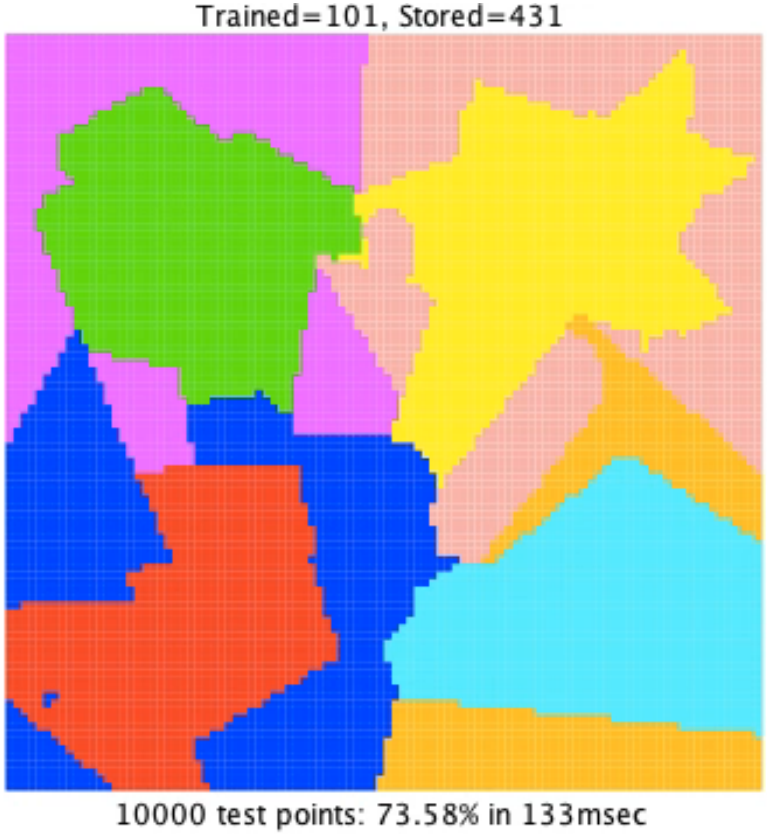
Improving Accuracy via Forests

Linear increase in memory + time

1 Tree



10 Trees



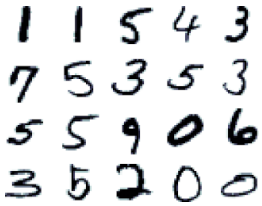
Classification Results

MNIST (60k training, 10k testing, 784 pixels)

Wall Clock Time (seconds)

	Training	Testing	Total
BF(50, 50)	103	2.3	**105.3**
1-NN	0	2900	2900.0
3-NN	0	3200	3200.0
RF(50, 50)	310	0.3	310.3

<2 msec train
<1 msec query



Error, Euclidean Distance

BF(1, 50)	1-CNN	RF(50, 50)	1-NN	3-NN	BF(50, 50)
12.15%	6.70%	3.16%	3.09%	2.83%	**2.32%**

Regression Results

YearPredictionMSD

- 463,715 (training) / 51,630 (testing)
- 90 features
- ~30x faster than 1-NN

RMSE, Euclidean Distance

1-NN	3-NN	BF(50, 50)
14.05	11.59	**10.41**

Possible Research Directions in Soar

Real-time learning of…

- perceptual patterns (e.g. color classification)
- action models (e.g. motion regression)
- long-term perceptual memories (via aNN)

Evaluation

- Fast & online algorithm that's easy to code/ understand
- Good performance on classification, regression, a-NN retrieval
- Many potential applications

- Needs a metric; little exploration of dynamic distance functions
- No work yet studying structured/temporal representations
- Future: incorporating dynamic priors

Thank You :)

Questions?

Disney Research

Algorithm Sketch
Required Parameters

- n_t = number of trees
- k = maximum number of children
  - 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 e

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))