k-Nearest Neighbors

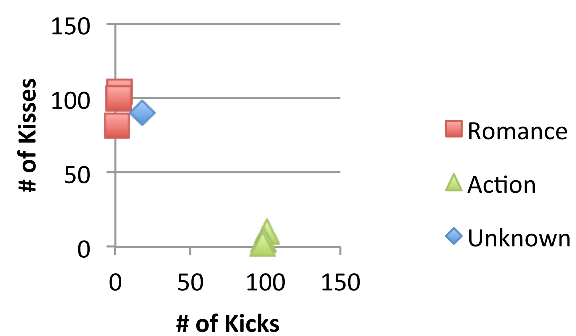
Lecture 2

Outline

1. Learning via distance measurements
2. Model parameters
   – Bias vs. Variance
3. Extensions
   – Regression
   – Improving Efficiency

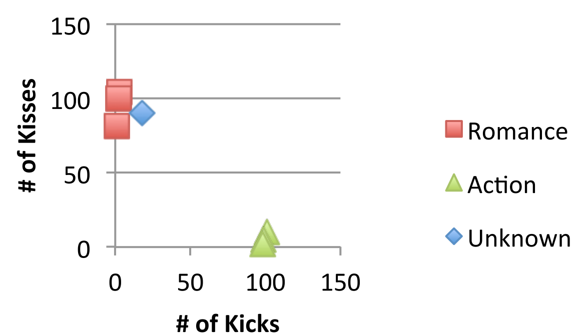
A Motivating Example

Movie Title	# of Kicks	# of Kisses	Type of Movie
California Man	3	104	Romance
He’s Not Really into Dudes	2	100	Romance
Beautiful Woman	1	81	Romance
Kevin Longblade	101	10	Action
Robo Slayer 3000	99	5	Action
Amped II	98	2	Action
?	**18**	**90**	**?**



A Motivating Example

Movie Title	# of Kicks	# of Kisses	Type of Movie	L2 Distance
California Man	3	104	Romance	20.52
He's Not Really into Dudes	2	100	Romance	18.87
Beautiful Woman	1	81	Romance	19.24
Kevin Longblade	101	10	Action	115.28
Robo Slayer 3000	99	5	Action	117.41
Amped II	98	2	Action	118.93
?	**18**	**90**	**?**	**0**



kNN

Training

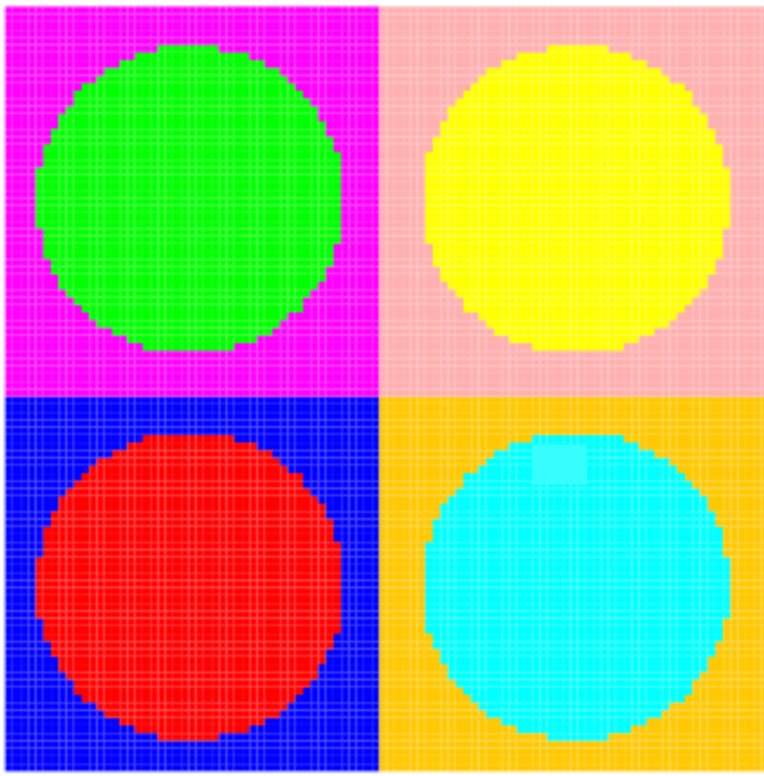
- Store examples

Testing

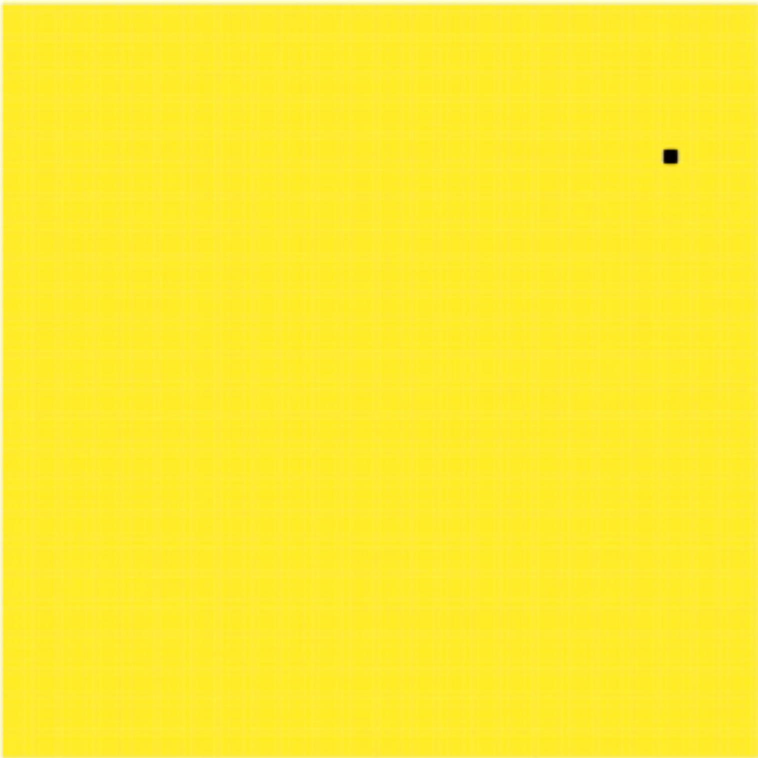
- Find the nearest *k* neighbors to target
  - Via distance function
- Vote on result

2D Multiclass Classification

Ground Truth



1-NN via Linear Scan



Model Parameters

- *k* – number of neighbors to find
- $D(\mathbf{x}_1,\mathbf{x}_2)$ – distance function
- $V(\{\mathbf{x}, y\})$ – voting function

<u>Related</u>

- Feature representation
  - Scaling
  - Curse of dimensionality
- Efficiency
  - Storage/search

Choosing *k*

- 1 = Nearest Neighbor

- Pro tip: if binary, choose odd to avoid ties

- Tradeoff: under/over-fitting
  - Small k: sensitive to noise
  - Large k: includes distal points

Bias vs. Variance Revisited

General

Model $y = f(x)$ as $\hat{f}(x)$

$$\mathrm{Err}(x) = \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Irreducible\ Error}$$

$$\mathrm{Err}(x) = E[(Y - \hat{f}(x))^2]$$

$$\mathrm{Bias} = E[\hat{f}(x)] - f(x)$$

$$\mathrm{Variance} = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$$

$$\mathrm{Irreducible\ Error} = \sigma^2$$

kNN

$$\mathrm{Bias} = f(x) - \frac{1}{k}\sum_{i=1}^{k} f(N_i(x))$$

Monotonically increases with *k*

$$\mathrm{Variance} = \frac{\sigma^2}{k}$$

Monotonically decreases with *k*

Example: http://scott.fortmann-roe.com/docs/BiasVariance.html

Common Distance Functions

- Manhattan (L1)
- Euclidean (L2)
- Cosine similarity
  - Useful in high dimensions: $cos(\theta) = \frac{A \cdot B}{||A|| \, ||B||}$
- Edit distance
- Graph traversal
  - Decay
- Modern: learn a useful distance measure!

Individual instance weighting

Issues with Distance Functions

- Categorical data
  - Indicator function is safe (i.e. Hamming Distance)
    - Pay attention to nominal features!
- Curses!
  - Euclidean becomes less discriminating in high dimensions
- Normalization
  - Consider a function over features
    - Annual salary
    - Height in meters
  - Common to scale features to [0, 1]

$$X_{\text{scaled}} = \frac{X - \text{Min}}{\text{Max} - \text{Min}}$$

V = Majority Vote

$$y' = \underset{v}{\operatorname{argmax}} \sum_{(\boldsymbol{x_i}, y_i) \in D_z} I(v = y_i)$$

V = Distance-Weighted Vote

$$y' = \operatorname*{argmax}_{v} \sum_{(\boldsymbol{x_i}, y_i) \in D_z} w_i \times I(v = y_i)$$

$$\text{where } w_i = \frac{1}{d(\boldsymbol{x}', \boldsymbol{x}_i)^2}$$

Useful if the nearest neighbors vary widely in their distance and the closer neighbors more reliably indicate the class of the object

Efficiency

Assume *N* training examples, *d* features…

- What is the computational cost of training a new instance?

$$\mathcal{O}(d) \sim \mathcal{O}(1)$$

- How much space is required to store the model?

$$\mathcal{O}(N \cdot d)$$

- What is the computational cost of predicting the result of a new test instance?

$$\mathcal{O}(N \cdot d)$$

Some Theory (Cover & Hart, 1967)

- ***Bayes error rate*** is the lowest possible error rate for a given class of classifier
  - Non-zero if the distributions of the instances overlap
  - More in later lectures
- As the amount of data approaches infinity, kNN is guaranteed to yield an error rate no worse than twice the Bayes error rate
- kNN is guaranteed to approach the Bayes error rate for some value of *k* (where *k* increases as a function of the number of data points)

Applying kNN to Regression

- Rather than voting on a label, the voting function produces a value
  - Average
  - Weighted average (w.r.t. distance)

Example: House Price Index

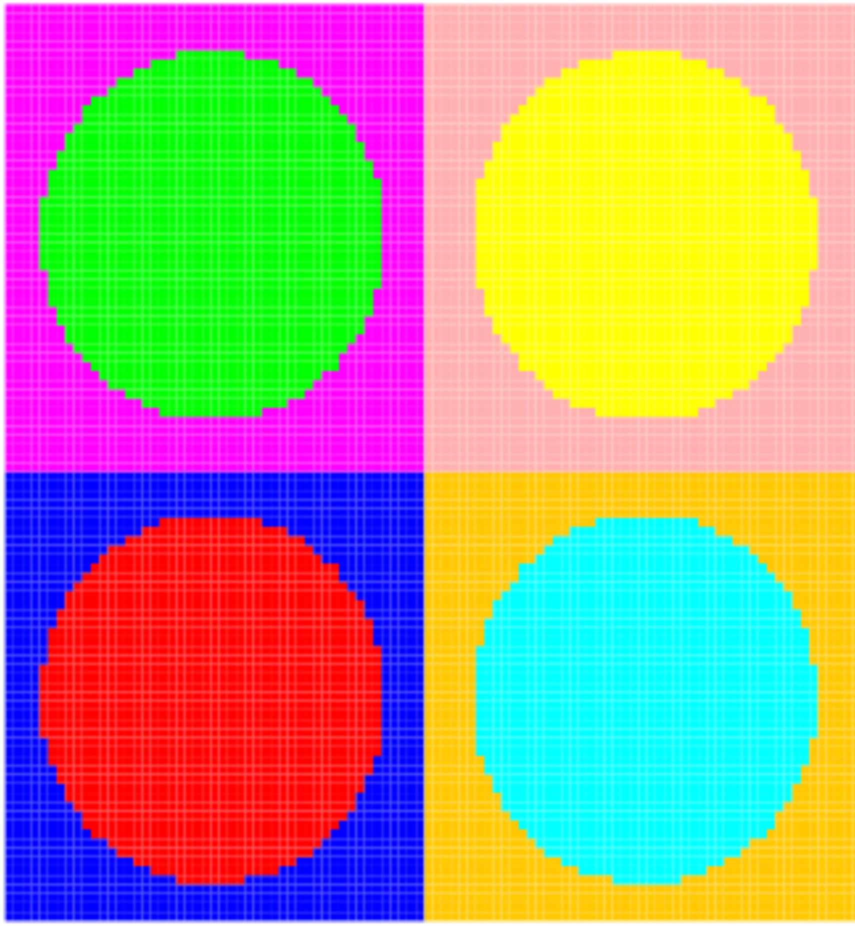
Age	Loan	House Price Index
25	$40,000	135
35	$60,000	256
45	$80,000	231
20	$20,000	267
35	$120,000	139
52	$18,000	150
23	$95,000	127
40	$62,000	216
60	$100,000	139
48	$220,000	250
33	$150,000	264
48	**$142,000**	**?**

http://www.saedsayad.com/k_nearest_neighbors_reg.htm

Improving Efficiency

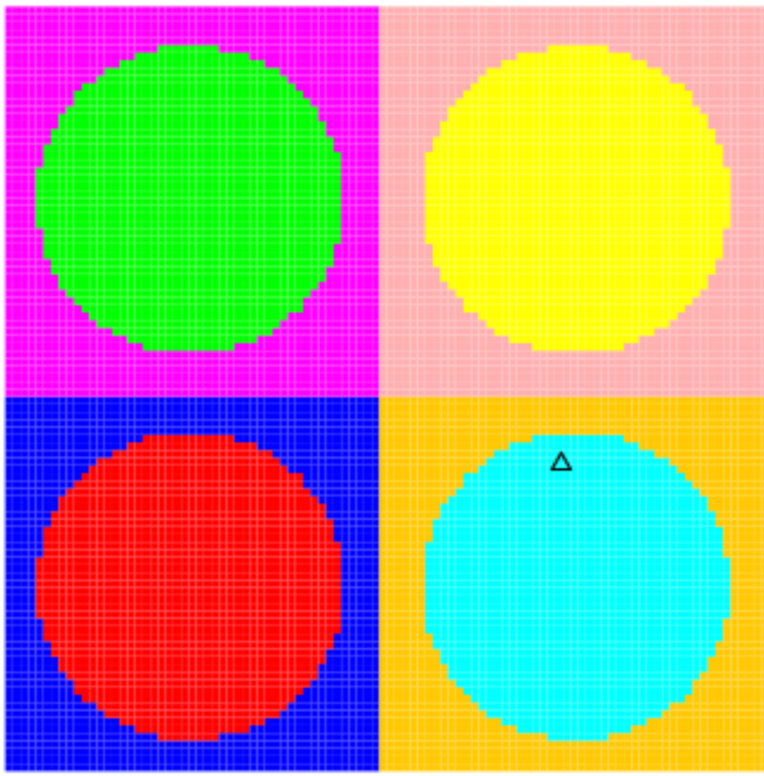
- Filtered Storage
  - Condensed NN
- Intelligent Search
  - Space partitioning (k-d tree, R-tree)
- Approximate NN
  - Locality Sensitive Hashing
  - **Boundary Forests**

A 2D Classification Example

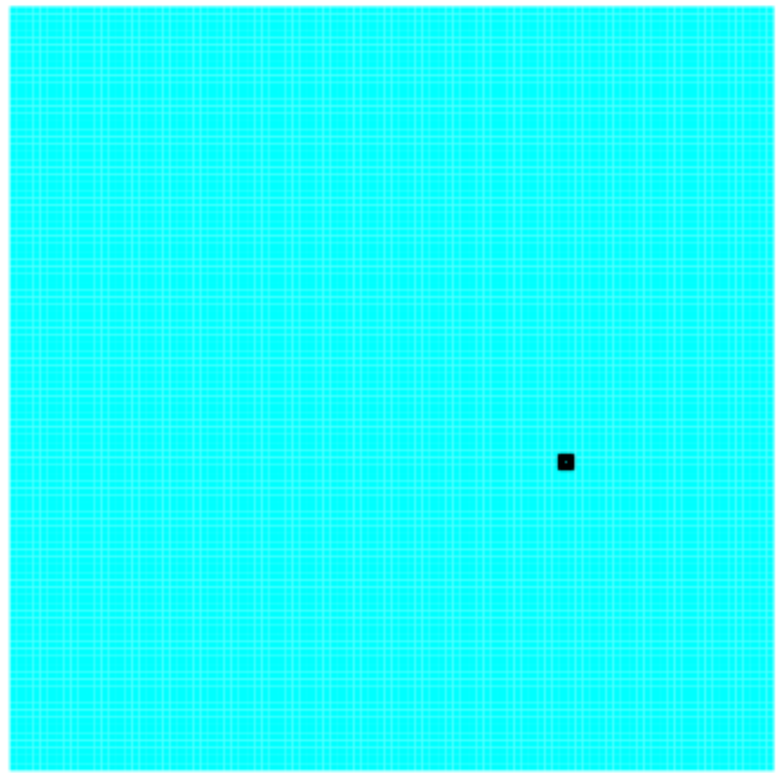


Interleaved Train/Query (1)

Ground Truth

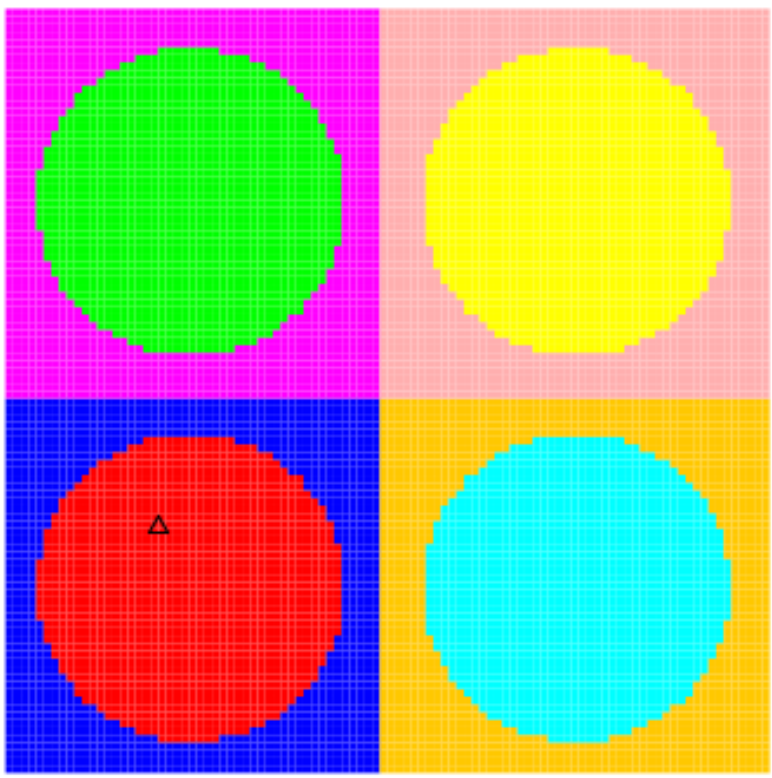


Boundary Tree

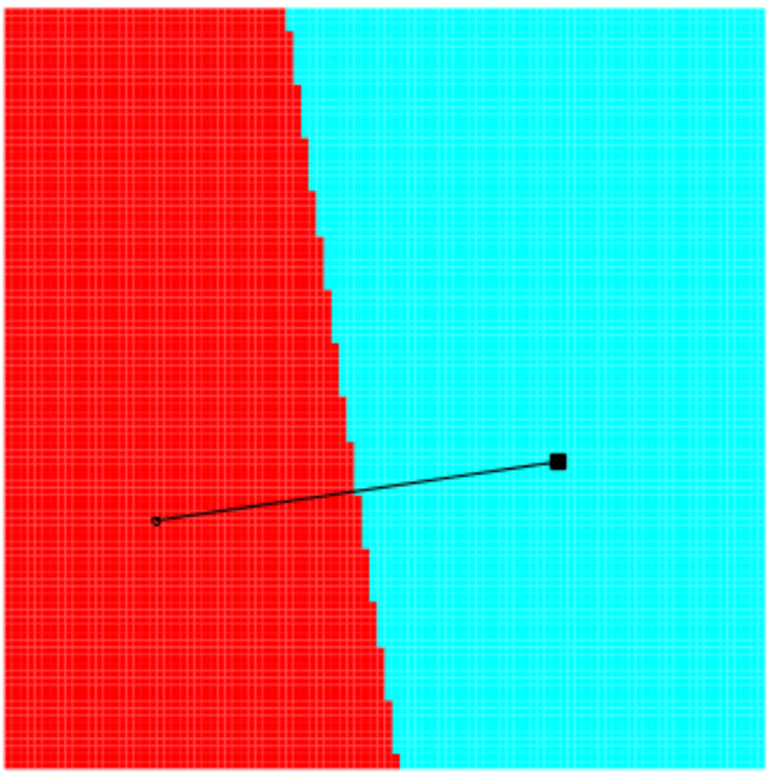


Interleaved Train/Query (2)

Ground Truth

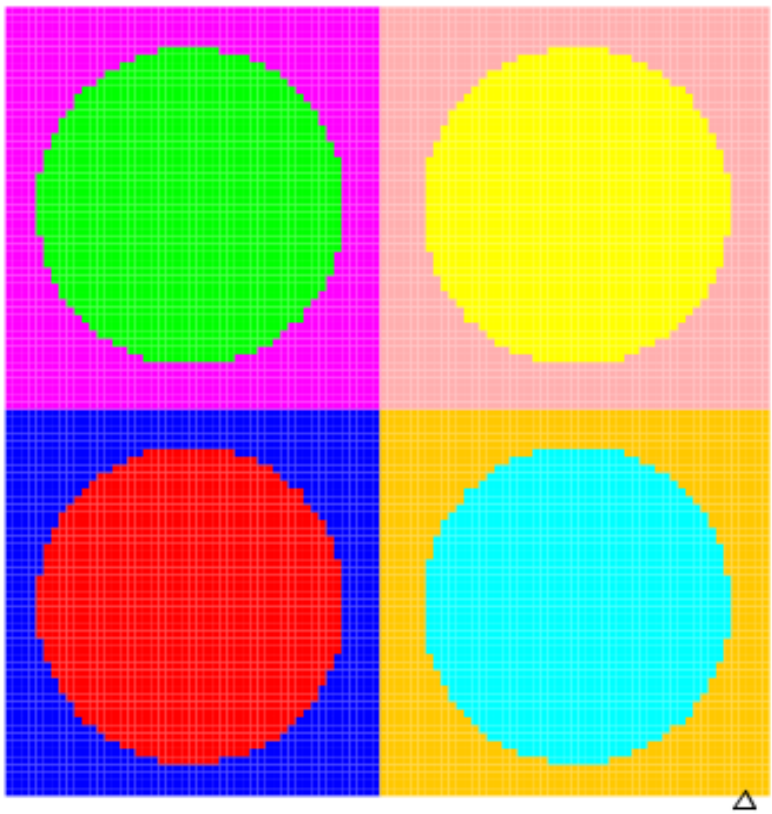


Boundary Tree

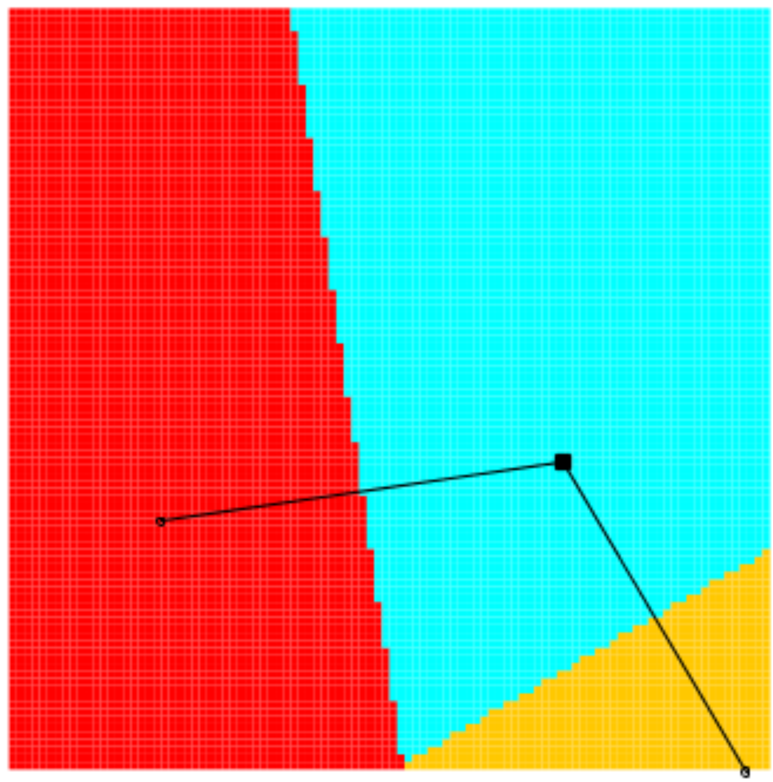


Interleaved Train/Query (3)

Ground Truth

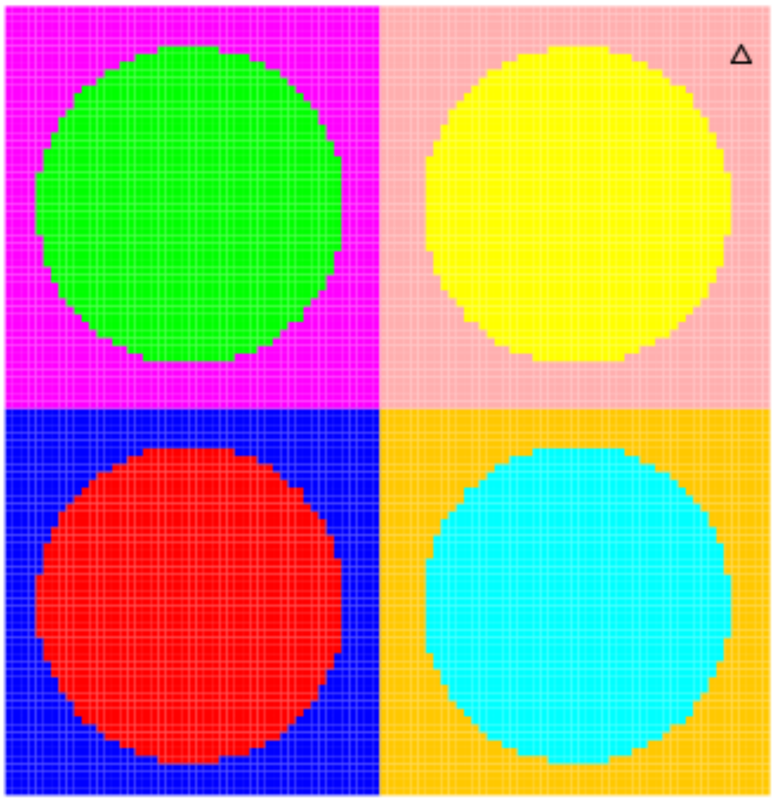


Boundary Tree

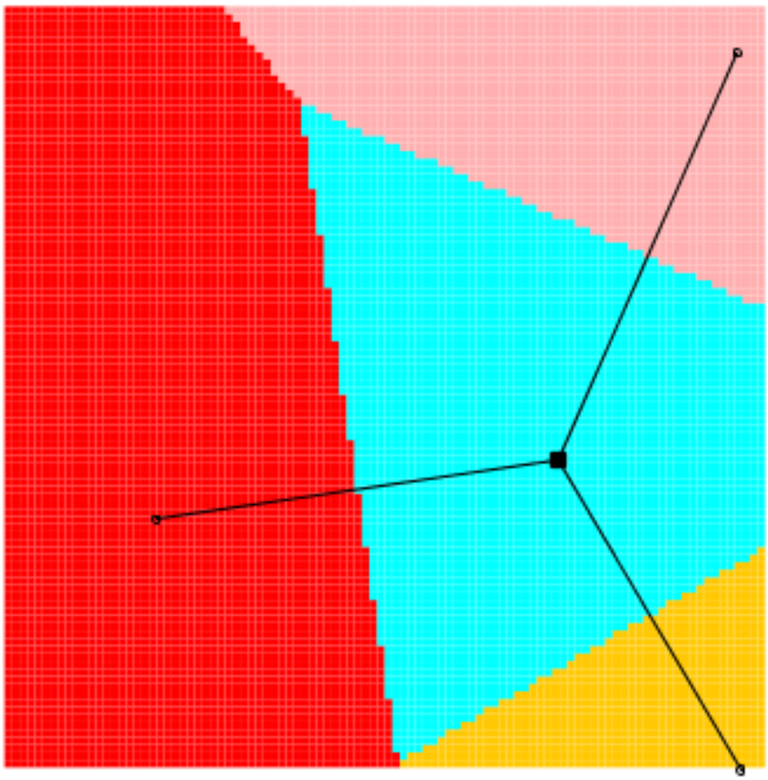


Interleaved Train/Query (4)

Ground Truth

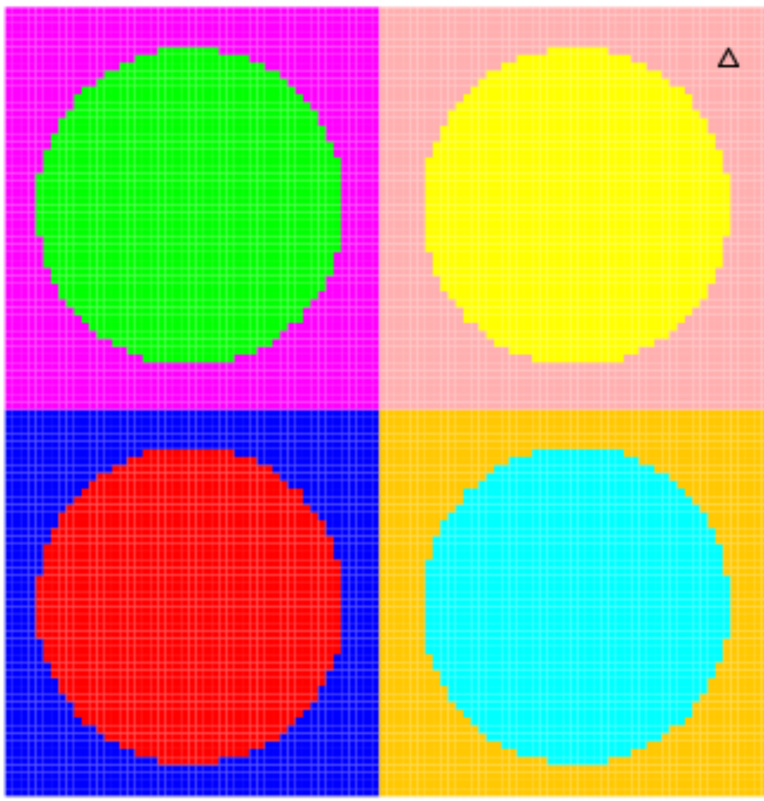


Boundary Tree

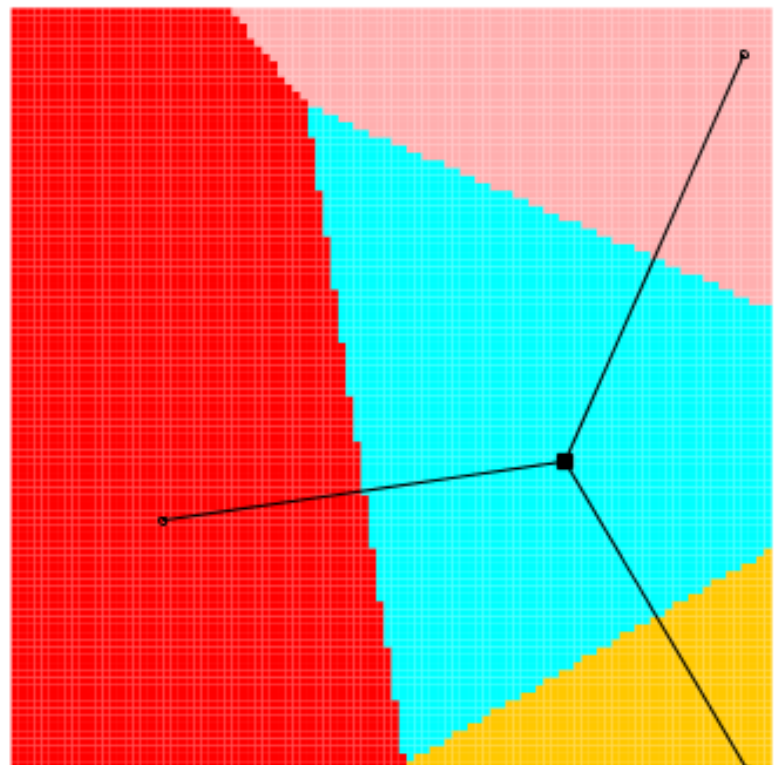


Interleaved Train/Query (5)

Ground Truth

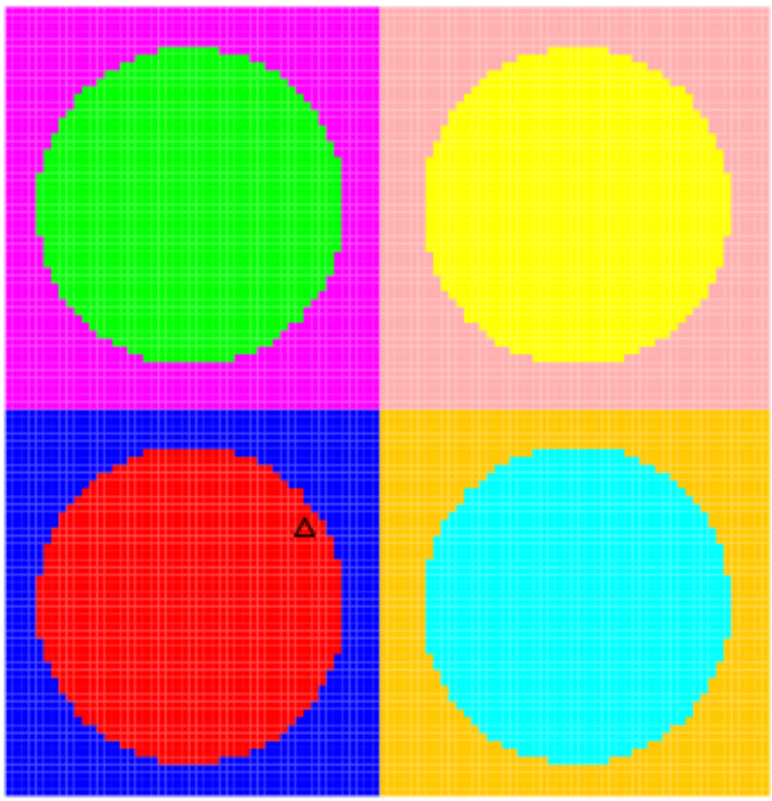


Boundary Tree

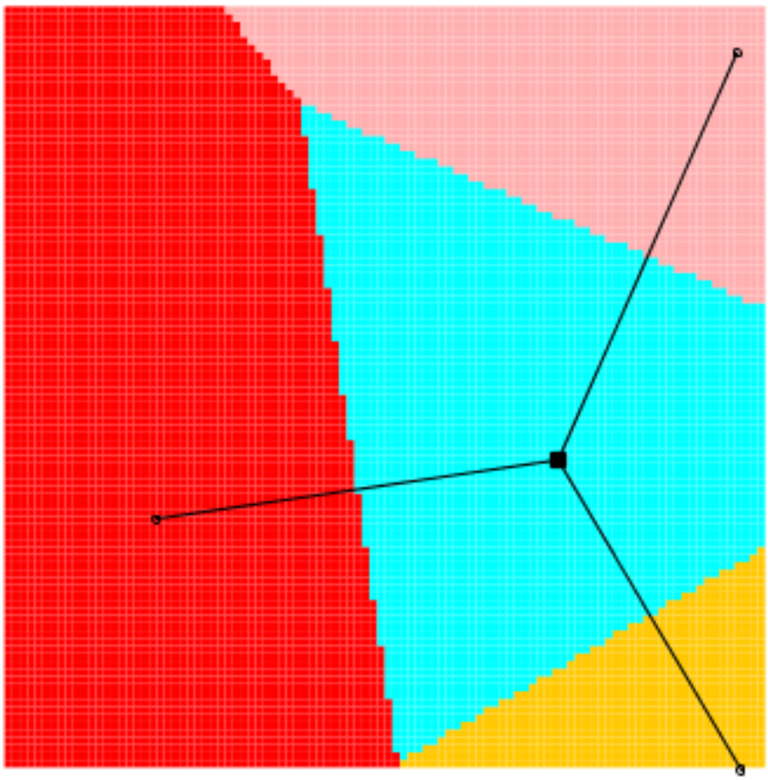


Interleaved Train/Query (6)

Ground Truth

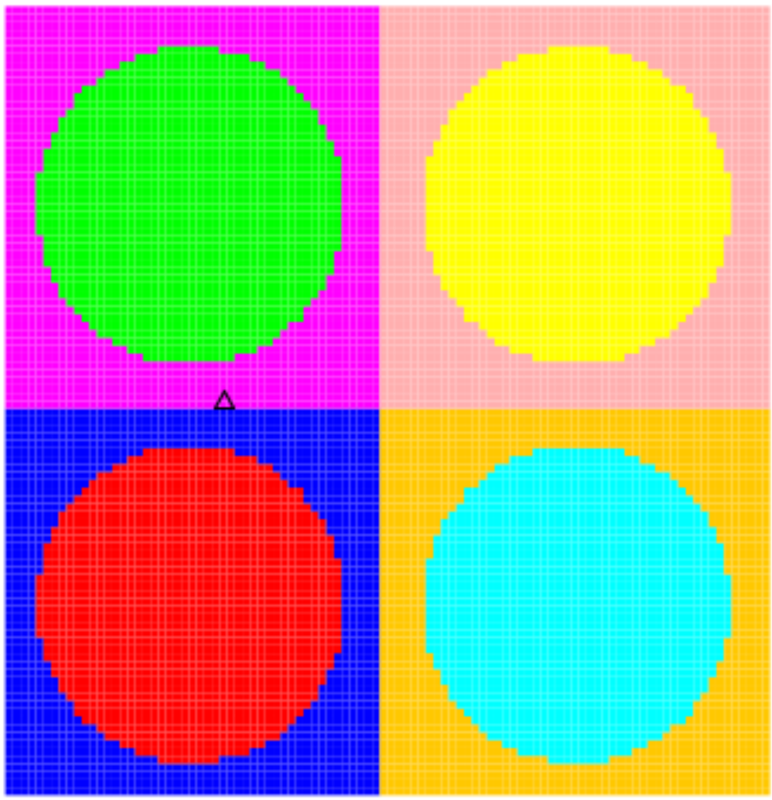


Boundary Tree

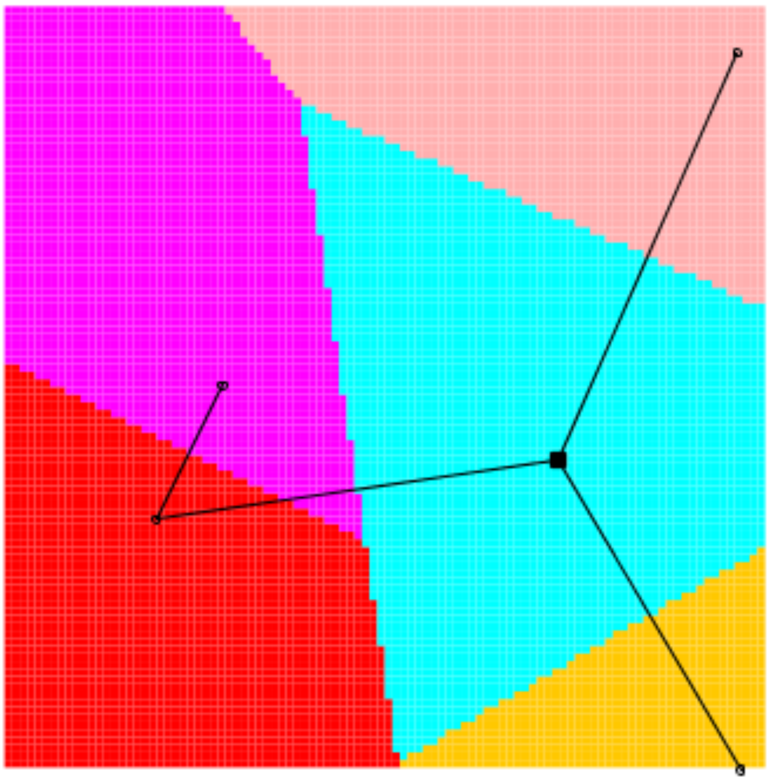


Interleaved Train/Query (7)

Ground Truth

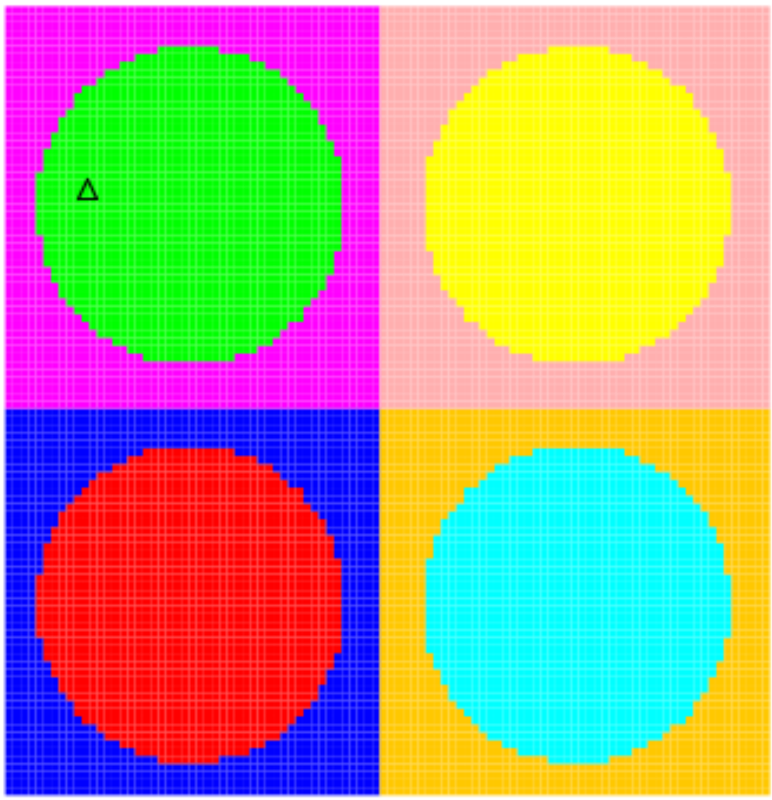


Boundary Tree

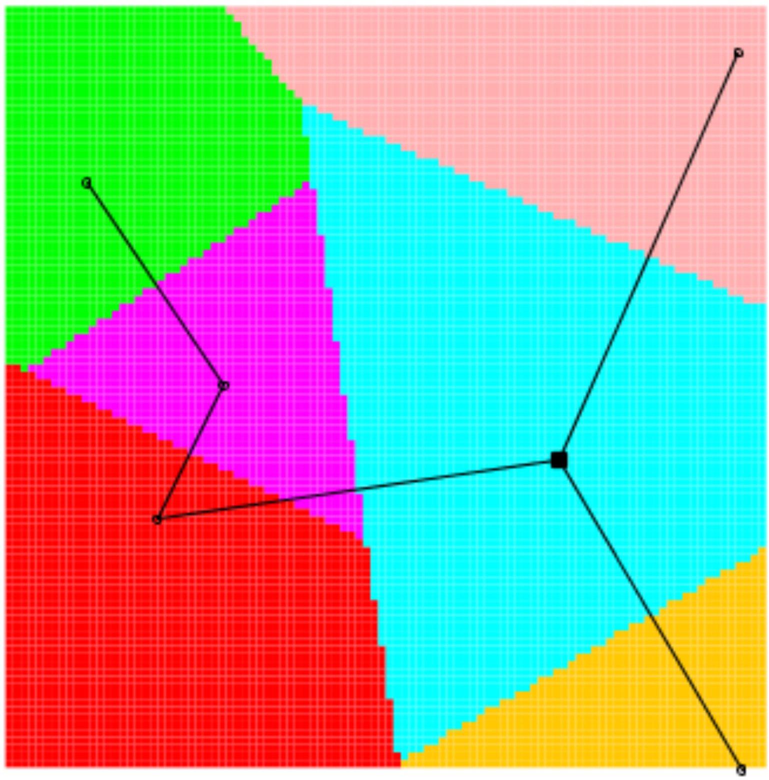


Interleaved Train/Query (8)

Ground Truth

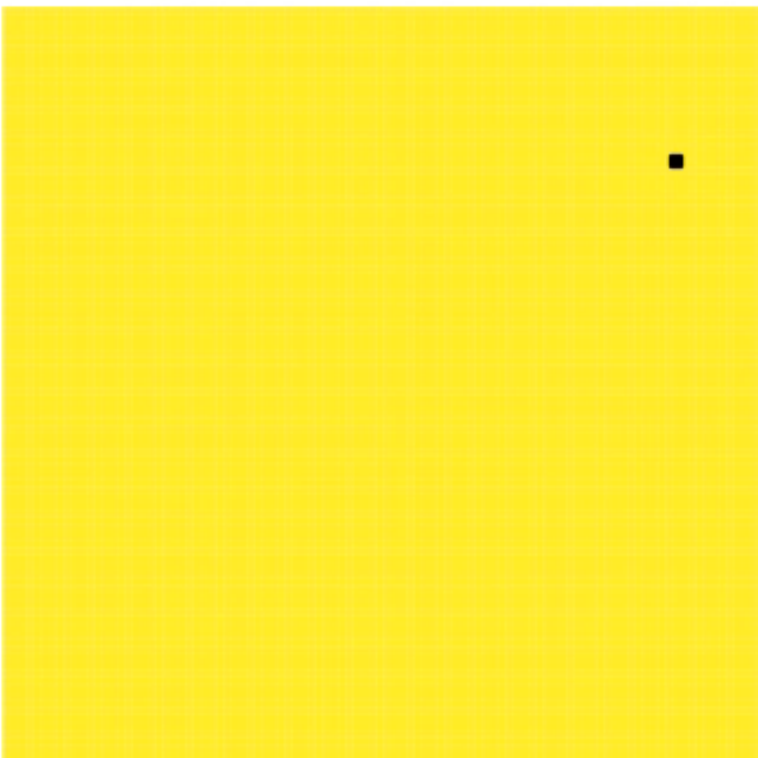


Boundary Tree

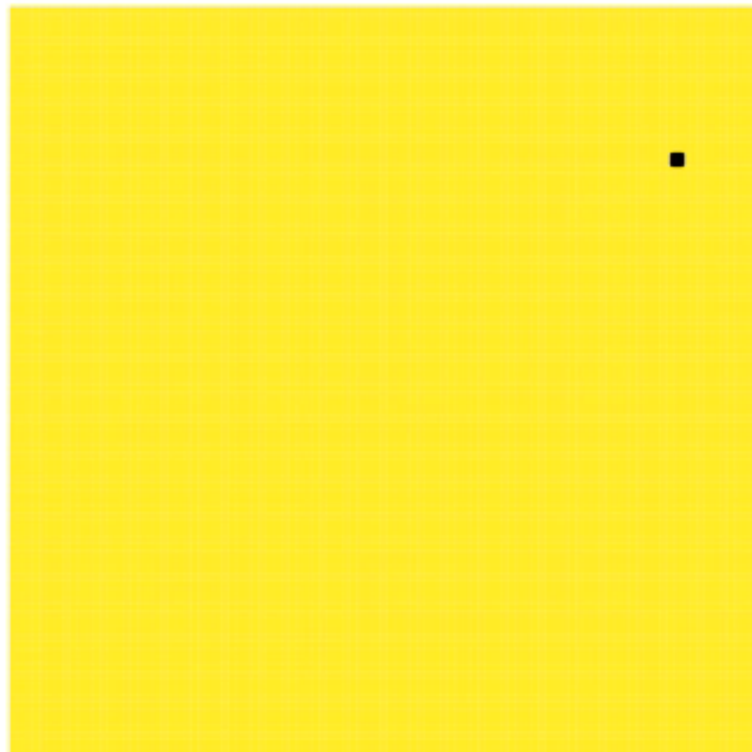


Performance & Scaling

Boundary Tree



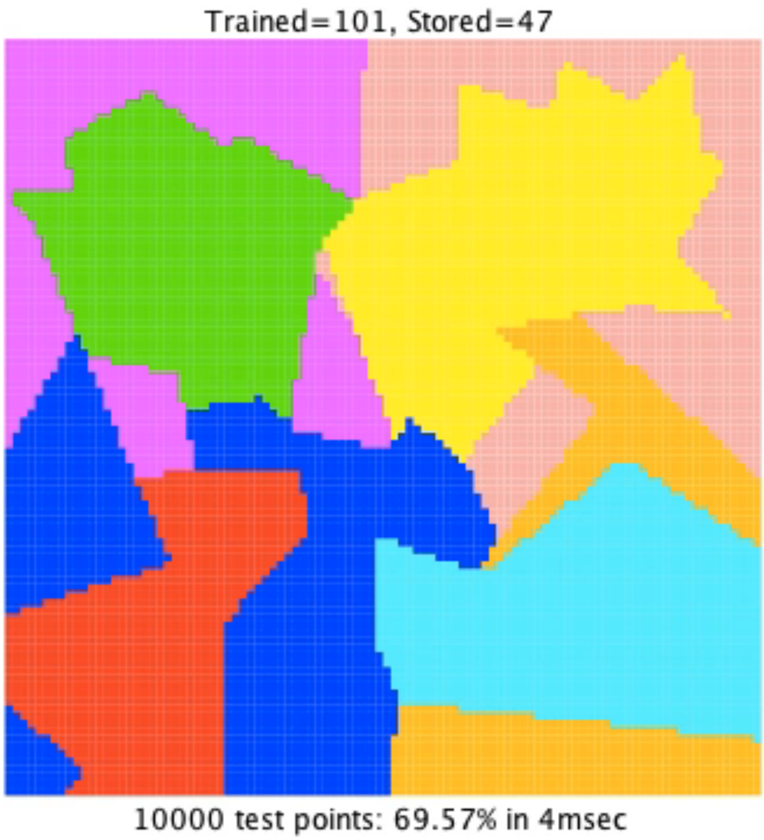
1-NN via Linear Scan



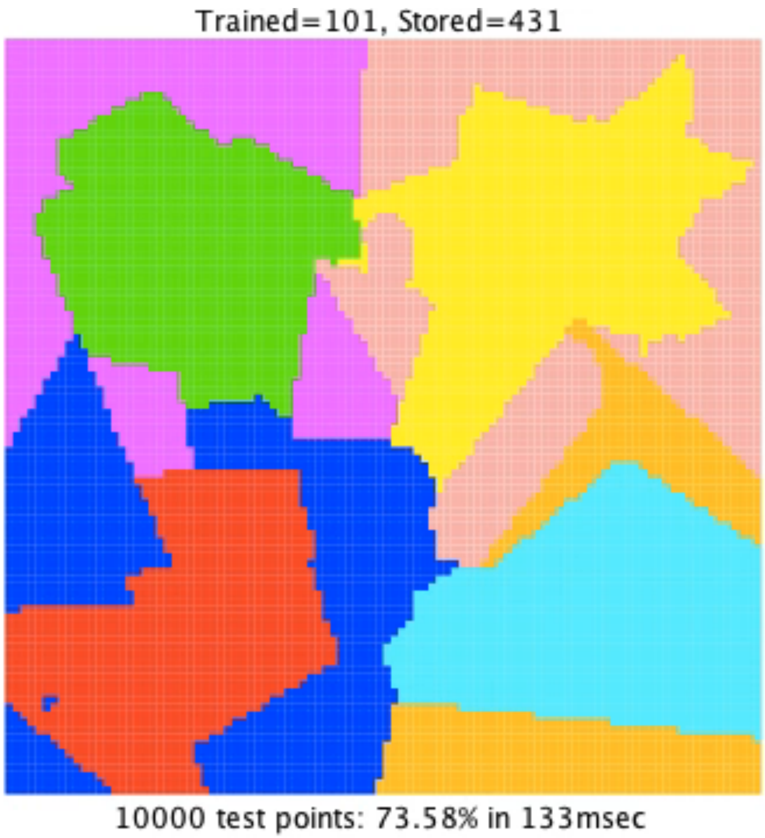
Improving Accuracy via Forests

Linear increase in memory + time

1 Tree



10 Trees



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;
  - $v = v_{min}$

Train(y)

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

<u>Result</u>

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

<u>ShouldAdd</u>

- 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))

Checkup

- ML task(s)?
  - Classification: binary/multi-class?
- Feature type(s)?
- Implicit/explicit?
- Parametric?
- Online?

Summary: kNN

- Practicality
  - Easy, generally applicable
  - Need know nothing about the underlying process
- Efficiency
  - Training: lazy
  - Testing: only for small datasets
    - Though there are methods to help scale
- Performance
  - Depends upon data/parameters (e.g. D, V, k, …)
  - Bounded above by twice the Bayes error under certain reasonable assumptions; the error of the general kNN method asymptotically approaches that of the Bayes error and can be used to approximate it