Machine Learning, Machine Learning (extended)

10 – Supervised Learning: Ensemble Methods

Kashif Rajpoot

k.m.rajpoot@cs.bham.ac.uk

School of Computer Science

University of Birmingham

Outline

- Ensemble methods
  - Boosting
  - Bagging
- Decision tree
- Random forests

Ensemble methods

- Combining 'weak classifiers' in order to produce a 'strong classifier'
  - "two heads are better than one"

- **Boosting**: train a new classifier focusing on training samples misclassified by an earlier classifier
  - Weak classifier: any classifier better than a random guess
  - AdaBoost

- **Bagging** (**b**ootstrap **agg**regation): generate new training data as a random subset of original data and train a new classifier on this subset
  - Weak classifier: a decision tree classifier
  - Random forests

Modified from Randomized Forests for Visual Recognition

Jamie Shotton — Microsoft Research Cambridge
Tae-Kyun Kim — University of Cambridge
Björn Stenger — Toshiba

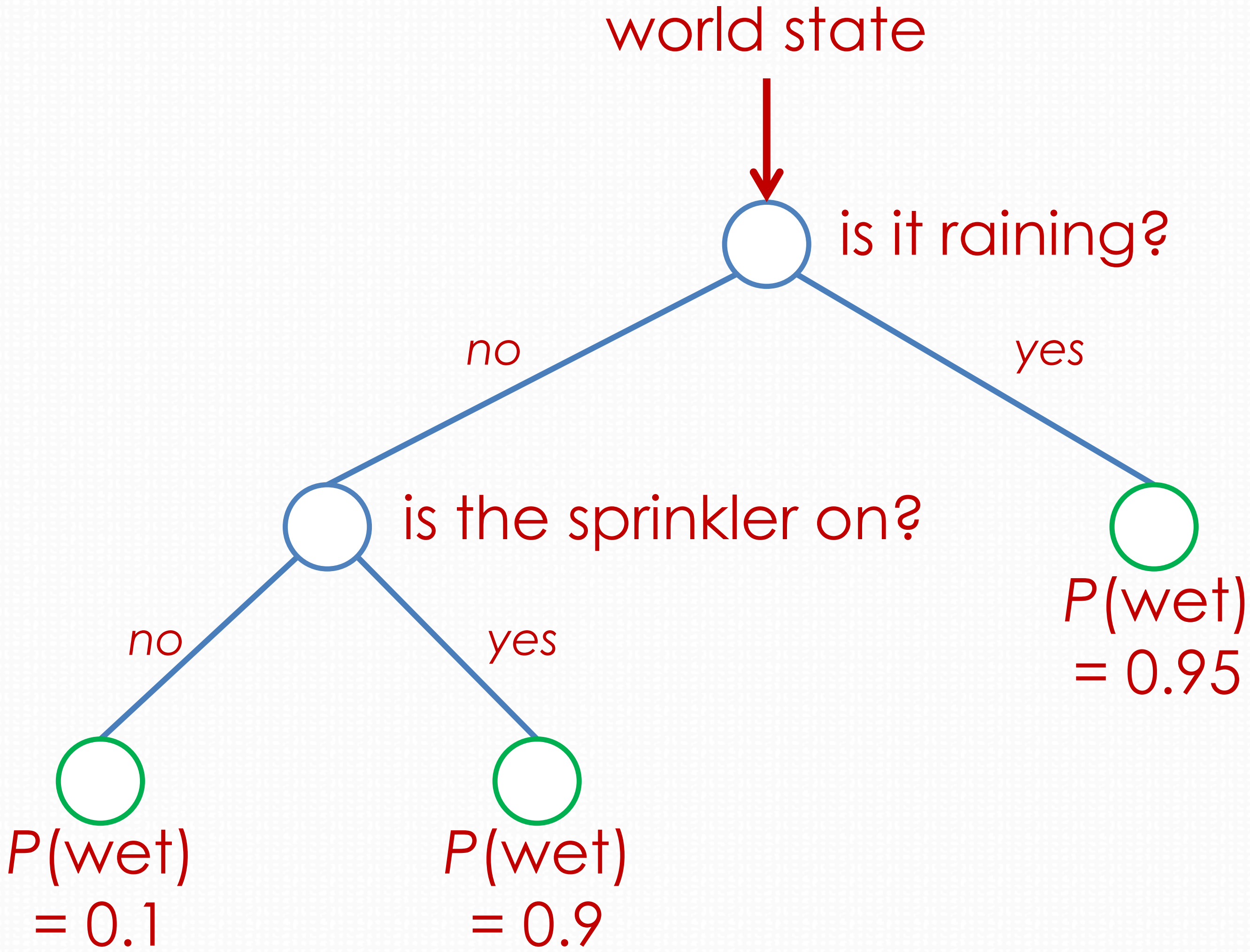
ICCV 2009, Kyoto, Japan

Randomized decision forests

- Very fast tool for classification
- Good generalization through randomized training
- Inherently multi-class
- Simple training / testing algorithms

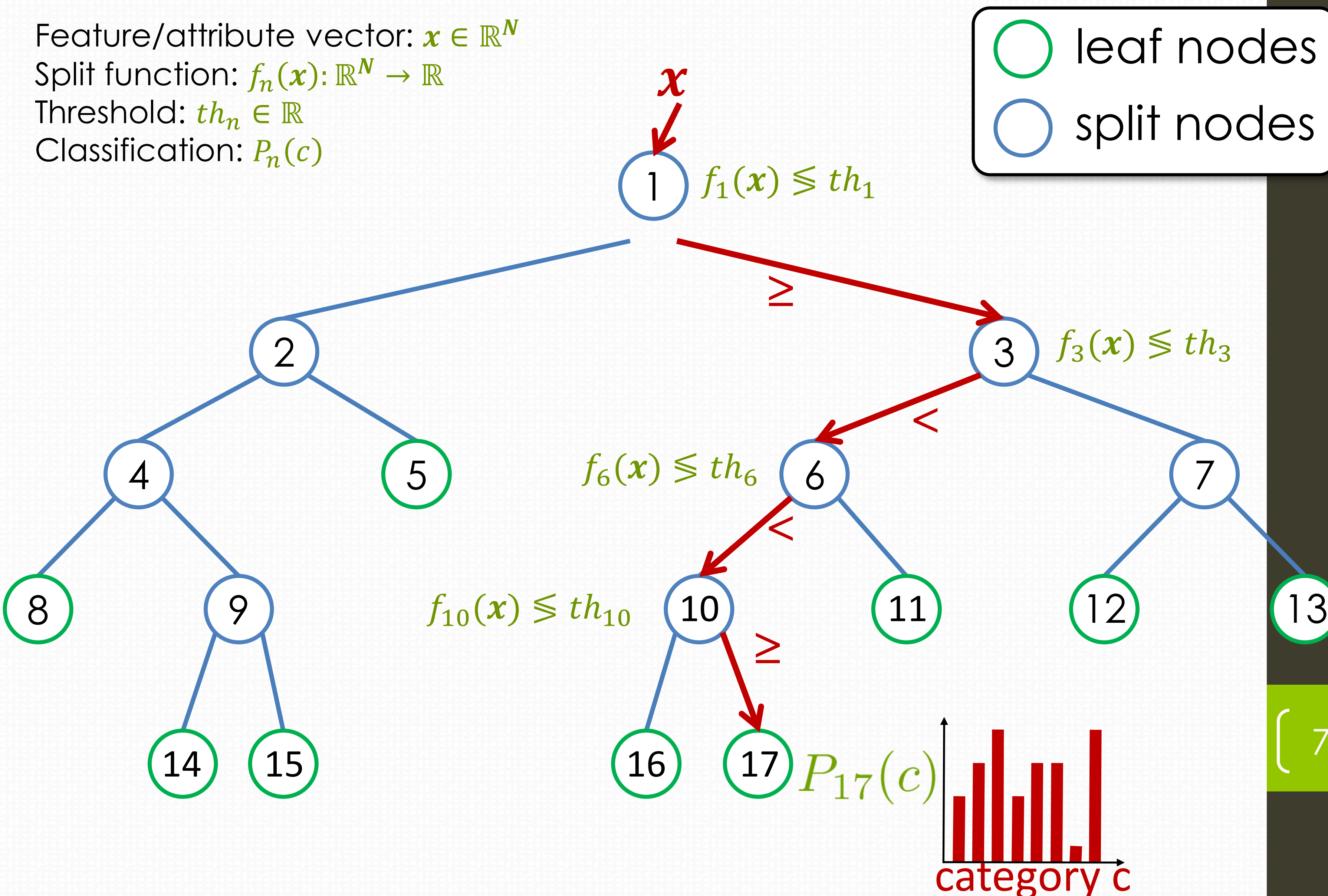
"Randomized Decision Forests" = "Randomized Forests" = "Random Forests™"

Basics: is the grass wet?



Basics: binary decision tree

Feature/attribute vector: $\boldsymbol{x} \in \mathbb{R}^N$
Split function: $f_n(\boldsymbol{x}): \mathbb{R}^N \rightarrow \mathbb{R}$
Threshold: $th_n \in \mathbb{R}$
Classification: $P_n(c)$

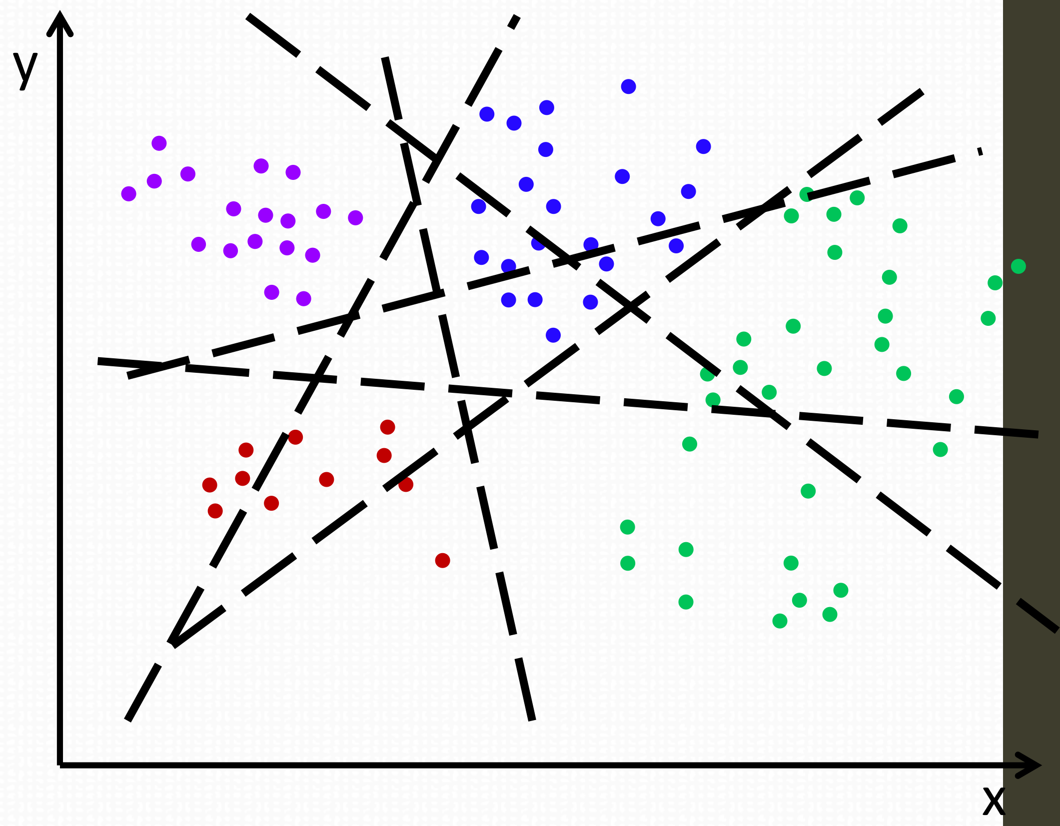


Decision tree classification: pseudo-code

```
double[] ClassifyDT(node, x)
    if node.IsSplitNode then
        if node.f(x) >= node.th then
            return ClassifyDT(node.right, x)
        else
            return ClassifyDT(node.left, x)
        end
    else
        return node.P
    end
end
```

Learning example

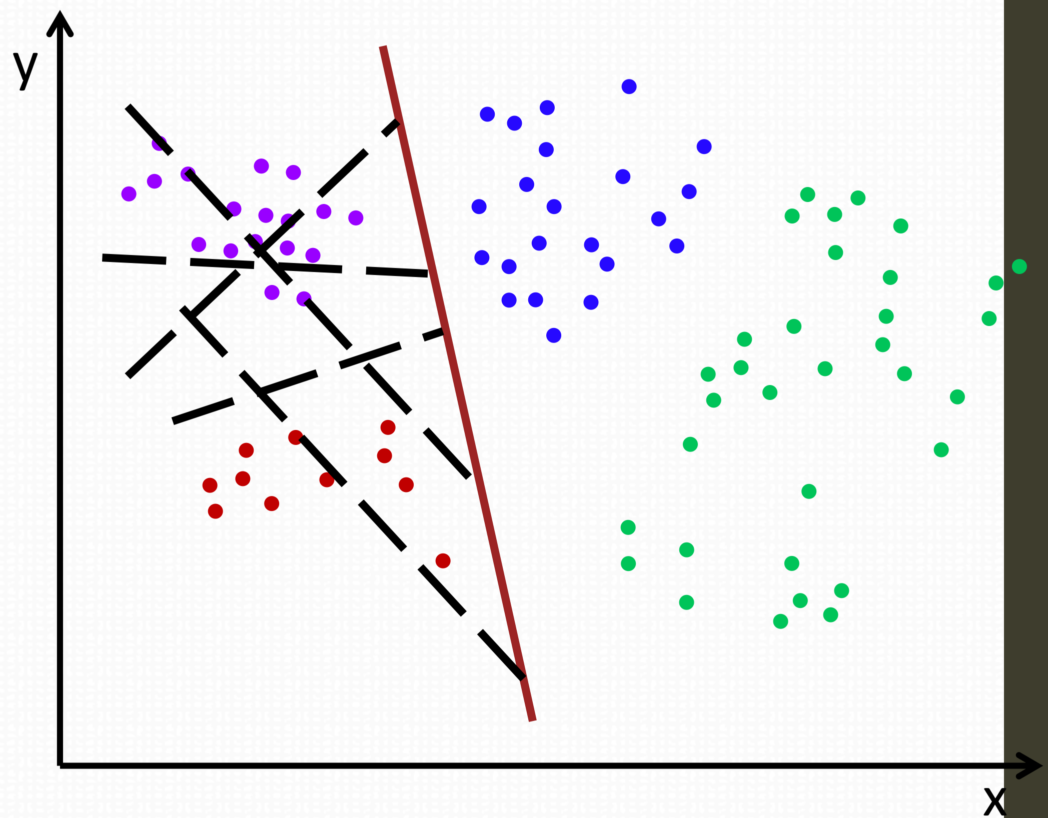
- Try several lines, 'chosen at random'
- Keep line that best separates data
  - Maximize information gain
- Recurse



- feature vectors are x, y coordinates: $\boldsymbol{x} = [x, y]^T$
- split functions are lines with parameters a, b: $f_n(\boldsymbol{x}) = ax + by$
- threshold determines intercepts: th_n
- four classes: purple, blue, red, green

Learning example

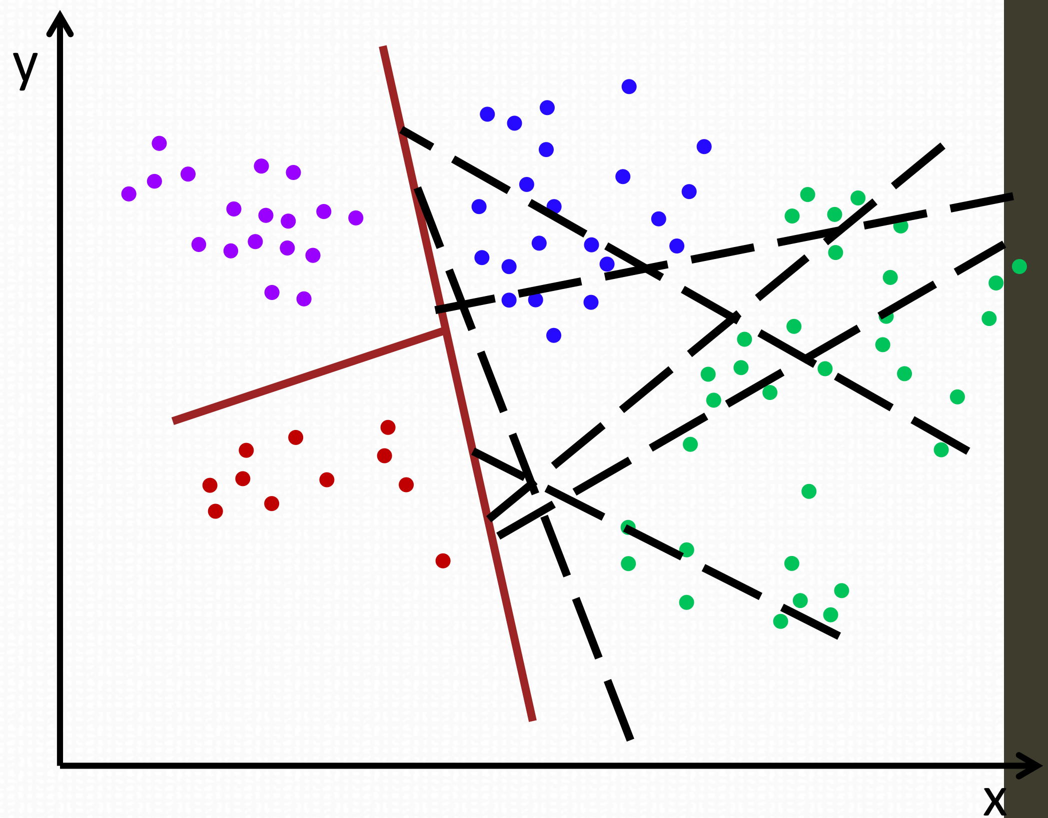
- Try several lines, 'chosen at random'
- Keep line that best separates data
  - Maximize information gain
- Recurse



- feature vectors are x, y coordinates: $\boldsymbol{x} = [x, y]^T$
- split functions are lines with parameters a, b: $f_n(\boldsymbol{x}) = ax + by$
- threshold determines intercepts: th_n
- four classes: purple, blue, red, green

Learning example

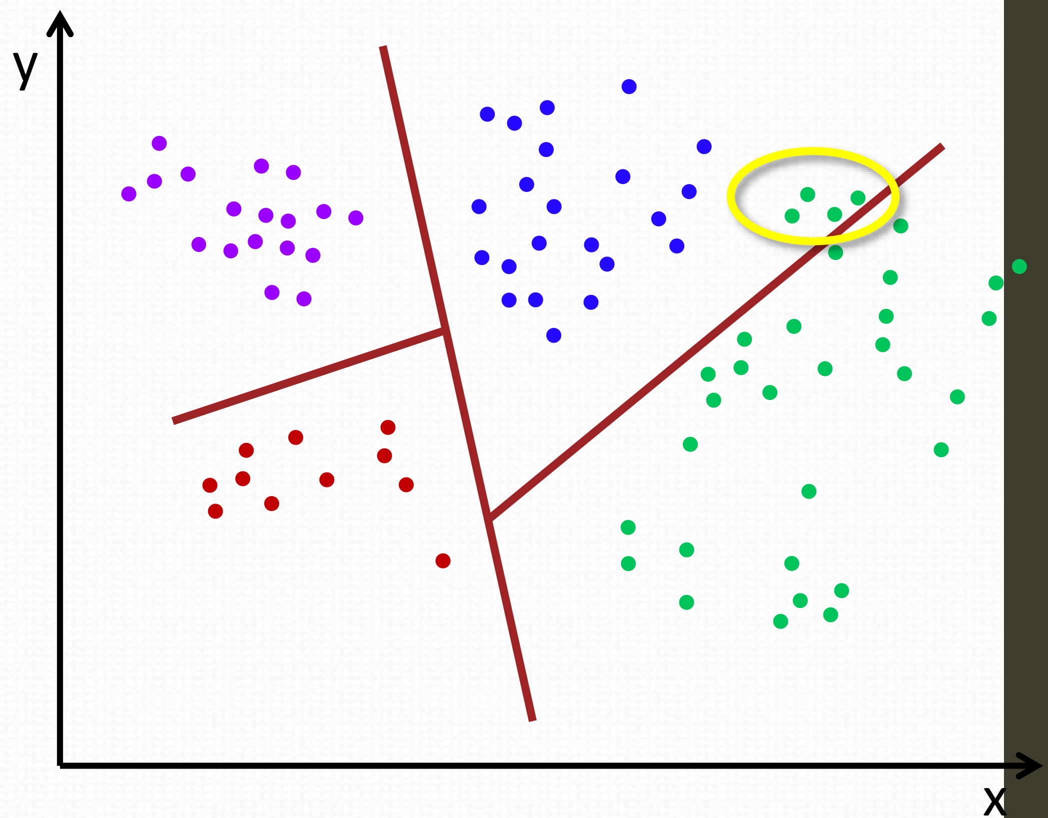
- Try several lines, 'chosen at random'
- Keep line that best separates data
  - Maximize information gain
- Recurse



- feature vectors are x, y coordinates: $\boldsymbol{x} = [x, y]^T$
- split functions are lines with parameters a, b: $f_n(\boldsymbol{x}) = ax + by$
- threshold determines intercepts: th_n
- four classes: purple, blue, red, green

Learning example

- Try several lines, 'chosen at random'
- Keep line that best separates data
  - Maximize information gain
- Recurse



- feature vectors are x, y coordinates: $\boldsymbol{x} = [x, y]^T$
- split functions are lines with parameters a, b: $f_n(\boldsymbol{x}) = ax + by$
- threshold determines intercepts: th_n
- four classes: purple, blue, red, green

Randomized learning

- Randomness in:
  - Bagging: randomly select the subset of data at the root of a tree
  - Randomly select the features at a tree node
    - "feature = attribute" in machine learning
  - Randomly select the threshold value

Randomized learning

- Randomly select X_n examples as a subset from X examples
- Recursively split X_n examples at node n

left split

$$X_l = \{\boldsymbol{x}_i \in X_n | f(\boldsymbol{x}_i) < th\}$$

threshold

$$X_r = \{\boldsymbol{x}_i \in X_n | f(\boldsymbol{x}_i) \geq th\}$$

right split

function of example i's feature vector

Randomized learning

- Features $f(\boldsymbol{x})$ chosen at random from feature pool F

- Threshold th chosen at random in range
  - $th \in \left(\min\left(f(\boldsymbol{x})\right), \max\left(f(\boldsymbol{x})\right)\right)$

- Choose f and th to maximize an objective function (e.g. information gain, Gini index)
  - Estimates whether it's "good" to distribute data further

Randomized learning

- $P_n(c)$ is the histogram (i.e. count) of example labels of class c which reached node n
- For example, at a leaf node, if 200 training example reach and there are 3 classes with following count, then the $P_n(c)$ is estimated as:

	$c = 1$	$c = 2$	$c = 3$
Count of examples	12	134	54
$P_n(c)$	$12/200$	134/200	$54/200$
$P_n(c)$	0.06	0.67	0.27

Implementation details

- How many features and thresholds to try?
  - just one = "extremely randomized"
  - few –> fast training, may under-fit
  - many –> slower training, may over-fit
- When to stop growing the tree?
  - maximum depth
  - minimum information gain

Decision tree learning: pseudo-code

```
TreeNode LearnDT(X)
   repeat featureTests times
      let f = RndFeature()
      let r = EvaluateFeatureResponses(X, f)

      repeat threshTests times
         let th = RndThreshold(r)
         let (X_l, X_r) = Split(X, r, th)

         let gain = InfoGain(X_l, X_r)
         if gain is best then remember f, th, X_l, X_r
      end
   end

   if best gain is sufficient
      return SplitNode(f, th, LearnDT(X_l), LearnDT(X_r))
   else
      return LeafNode(HistogramExamples(X_s))
   end
end
```

Binary decision tree: summary

- Fast greedy training algorithm
  - can search infinite pool of features
  - heterogeneous pool of features

- Fast testing algorithm

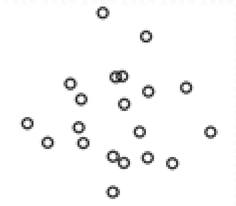
- Needs careful choice of hyper-parameters
  - maximum depth
  - number of features and thresholds to try

Information gain and entropy

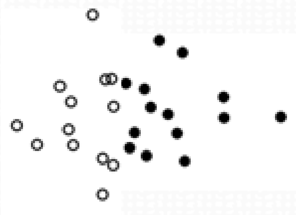
- Information gain: gain in information (i.e purity of data according to class labels) by split of data from parent to child nodes in the tree
  - $IG(f_n) = E(parent) - \frac{|X_l|}{|X_n|} * E(left) - \frac{|X_r|}{|X_n|} * E(right)$

  where $|X_n|$ denotes number of data samples at node n

- Entropy: measure of disorder (or impurity) in a bunch of data samples
  - $E = -\sum_{c=1}^{C} P_c \log_2(P_c)$



Low entropy



High entropy

Information gain and entropy

- $E = -\sum_{c=1}^{C} P_c \log_2(P_c) = ?$
  - $E = -P_{Flu=Y} \log_2(P_{Flu=Y}) - P_{Flu=N} \log_2(P_{Flu=N}) = ?$
  - $E = -\frac{5}{8} \log_2\left(\frac{5}{8}\right) - \frac{3}{8} \log_2\left(\frac{3}{8}\right) = 0.9544$

chills	runny nose	headache	fever	Flu?
Y	N	Mild	Y	N
Y	Y	No	N	Y
Y	N	Strong	Y	Y
N	Y	Mild	Y	Y
N	N	No	N	N
N	Y	Strong	Y	Y
N	Y	Strong	N	N
Y	Y	Mild	Y	Y

Information gain and entropy

$E(left) = ?$

$E(chills = Y) = -P_{Flu=Y}\log_2(P_{Flu=Y}) - P_{Flu=N}\log_2(P_{Flu=N}) = ?$

$E(chills = Y) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) = 0.8113$

- $IG(chills) = E(parent) - \frac{|X_l|}{|X_n|} * E(left) - \frac{|X_r|}{|X_n|} * E(right) = ?$
- $IG(chills) = 0.9544 - 0.5 * 0.8113 - 0.5 * 1 = 0.0488$

$E(right) = ?$

$E(chills = N) = -P_{Flu=Y}\log_2(P_{Flu=Y}) - P_{Flu=N}\log_2(P_{Flu=N}) = ?$

$E(chills = N) = -\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log_2\left(\frac{2}{4}\right) = 1$

$\frac{|X_l|}{|X_n|} = ?$

$\frac{|X_{chills=Y}|}{|X_n|} = ?$

$\frac{|X_{chills=Y}|}{|X_n|} = \frac{4}{8}$

$\frac{|X_r|}{|X_n|} = ?$

$\frac{|X_{chills=N}|}{|X_n|} = ?$

$\frac{|X_{chills=N}|}{|X_n|} = \frac{4}{8}$

chills	runny nose	headache	fever	Flu?
Y	N	Mild	Y	N
Y	Y	No	N	Y
Y	N	Strong	Y	Y
N	Y	Mild	Y	Y
N	N	No	N	N
N	Y	Strong	Y	Y
N	Y	Strong	N	N
Y	Y	Mild	Y	Y

Information gain and entropy

$E(left) = ?$

$E(runny = Y) = -P_{Flu=Y}\log_2(P_{Flu=Y}) - P_{Flu=N}\log_2(P_{Flu=N}) = ?$

$E(runny = Y) = -\frac{4}{5}\log_2\left(\frac{4}{5}\right) - \frac{1}{5}\log_2\left(\frac{1}{5}\right) = 0.7219$

- $IG(runny) = E(parent) - \frac{|X_l|}{|X_n|} * E(left) - \frac{|X_r|}{|X_n|} * E(right) = ?$
- $IG(runny) = 0.9544 - 0.625 * 0.7219 - 0.375 * 0.9183 = 0.1589$

$E(right) = ?$

$E(runny = N) = -P_{Flu=Y}\log_2(P_{Flu=Y}) - P_{Flu=N}\log_2(P_{Flu=N}) = ?$

$E(runny = N) = -\frac{1}{3}\log_2\left(\frac{1}{3}\right) - \frac{2}{3}\log_2\left(\frac{2}{3}\right) = 0.9183$

$\frac{|X_l|}{|X_n|} = ?$

$\frac{|X_{runny=Y}|}{|X_n|} = ?$

$\frac{|X_{runny=Y}|}{|X_n|} = \frac{5}{8}$

$\frac{|X_r|}{|X_n|} = ?$

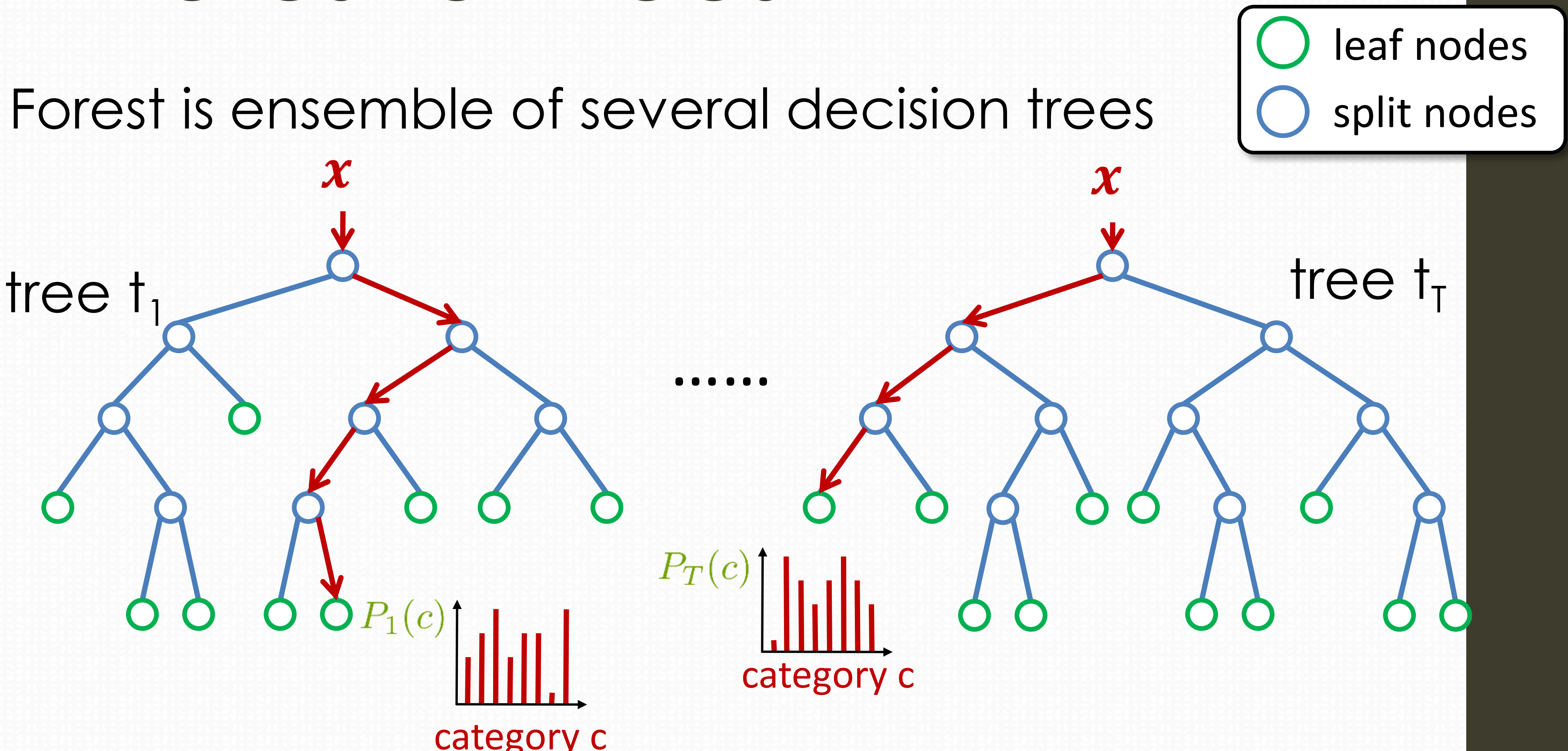
$\frac{|X_{runny=N}|}{|X_n|} = ?$

$\frac{|X_{runny=N}|}{|X_n|} = \frac{3}{8}$

chills	runny nose	headache	fever	Flu?
Y	N	Mild	Y	N
Y	Y	No	N	Y
Y	N	Strong	Y	Y
N	Y	Mild	Y	Y
N	N	No	N	N
N	Y	Strong	Y	Y
N	Y	Strong	N	N
Y	Y	Mild	Y	Y

A forest of trees

- Forest is ensemble of several decision trees



- Classification
  - $P(c|\boldsymbol{x}) = \frac{1}{T}\sum_{tr=1}^{T} P_{tr}(c|\boldsymbol{x})$

Decision forests: pseudo-code

```
double[] ClassifyDF(forest, x)
    // allocate memory
    let P = double[forest.CountClasses]

    // loop over trees in forest
    for tr = 1 to forest.CountTrees
        let P’ = ClassifyDT(forest.Tree[tr], x)
        P = P + P’ // sum distributions
    end

    // normalise
    P = P / forest.CountTrees
end
```

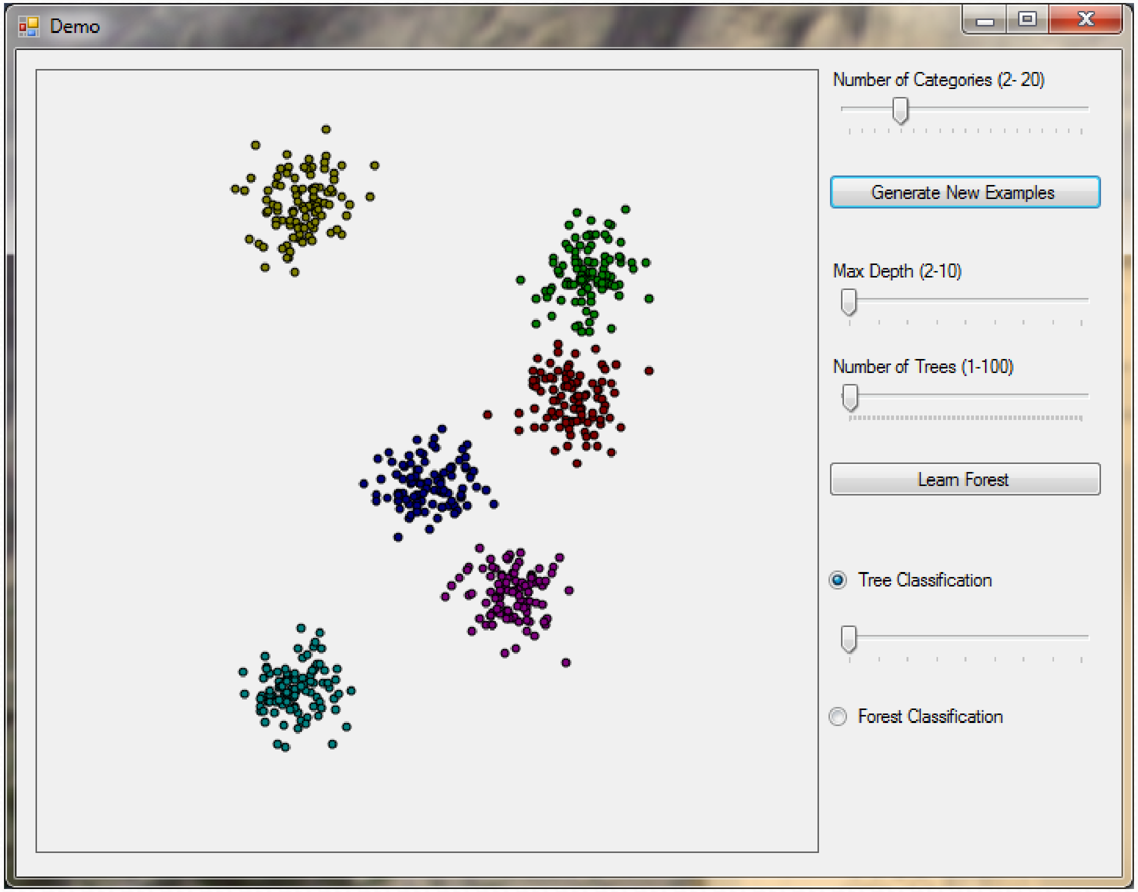
Learning a forest

- Divide training examples into T subsets
  - $X_{tr} \subseteq X$
  - improves generalization
  - reduces memory requirements & training time
- Subsets are chosen at random
- Subsets can have overlap (and usually do)
- Train each decision tree tr on subset X_{tr}

Learning a forest: pseudo-code

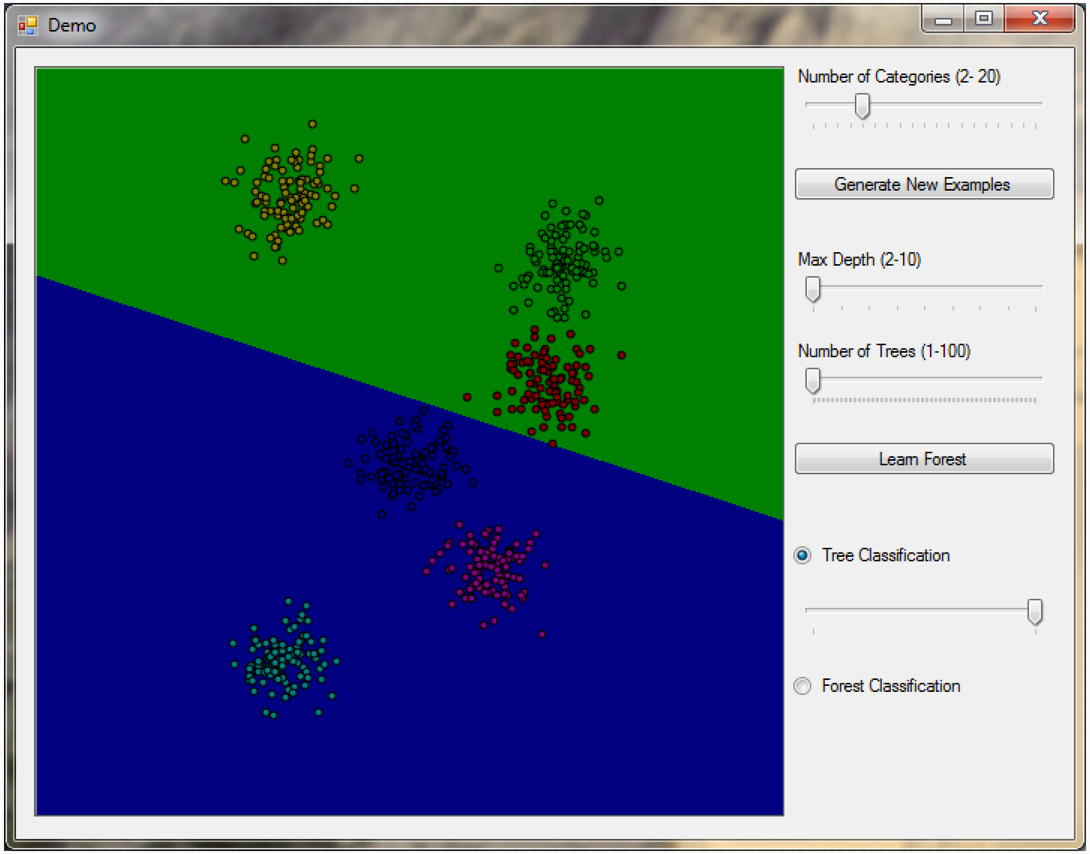
```
Forest LearnDF(countTrees, X)
    // allocate memory
    let forest = Forest(countTrees)

    // loop over trees in forest
    for tr = 1 to countTrees
        let X_tr = RandomSplit(X)
        forest[tr] = LearnDT(X_tr)
    end

    // return forest object
    return forest
end
```

Random forest: demo



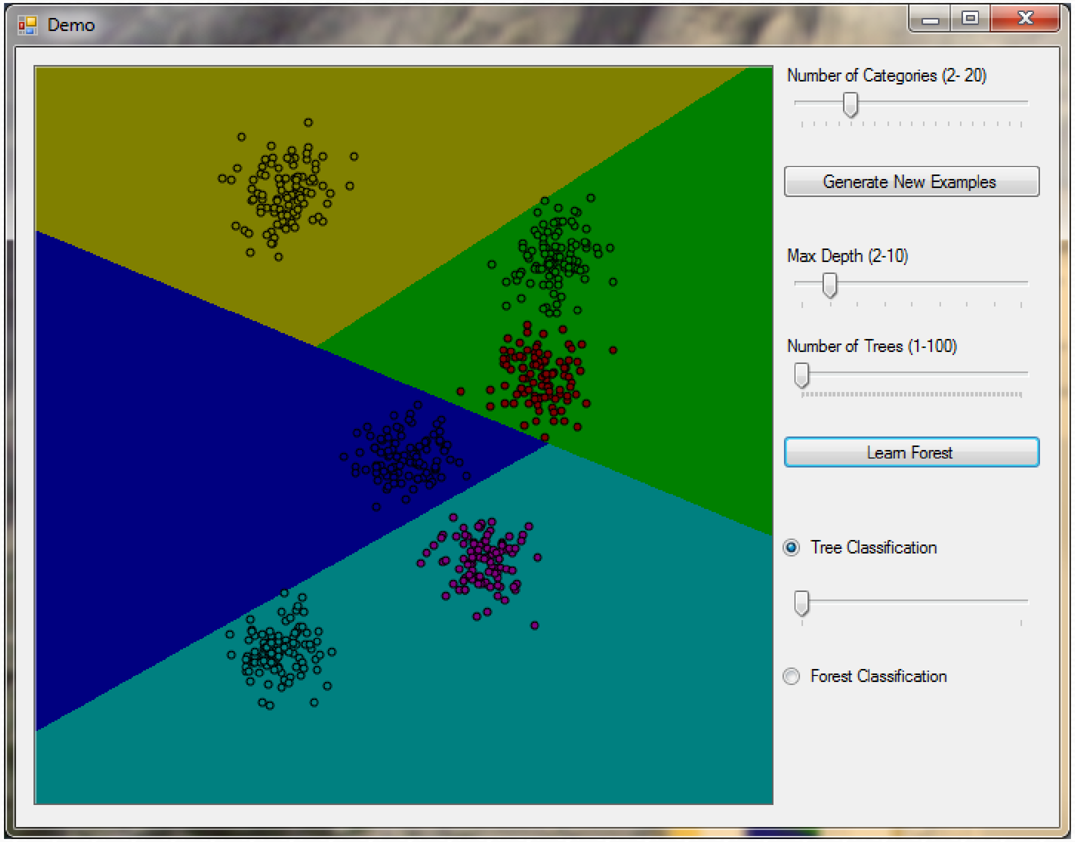
6 classes in a 2 dimensional feature space.
Split functions are lines in this space.

Random forest: demo



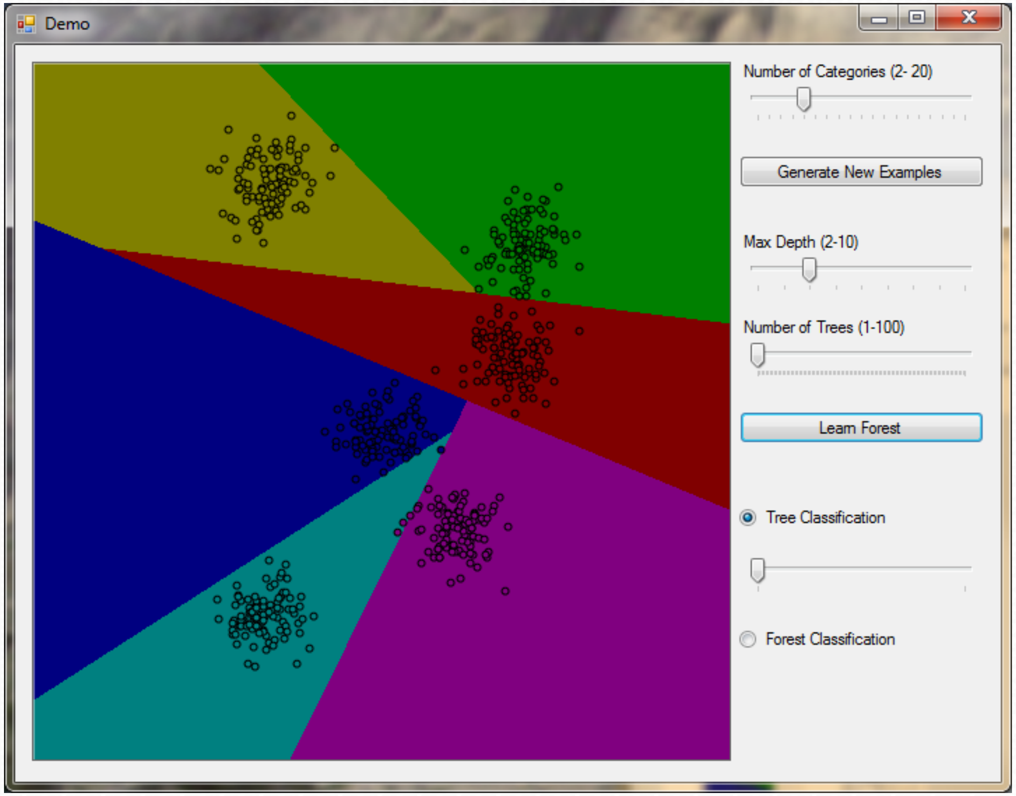
With a depth 2 tree, you cannot separate all six classes.

Random forest: demo



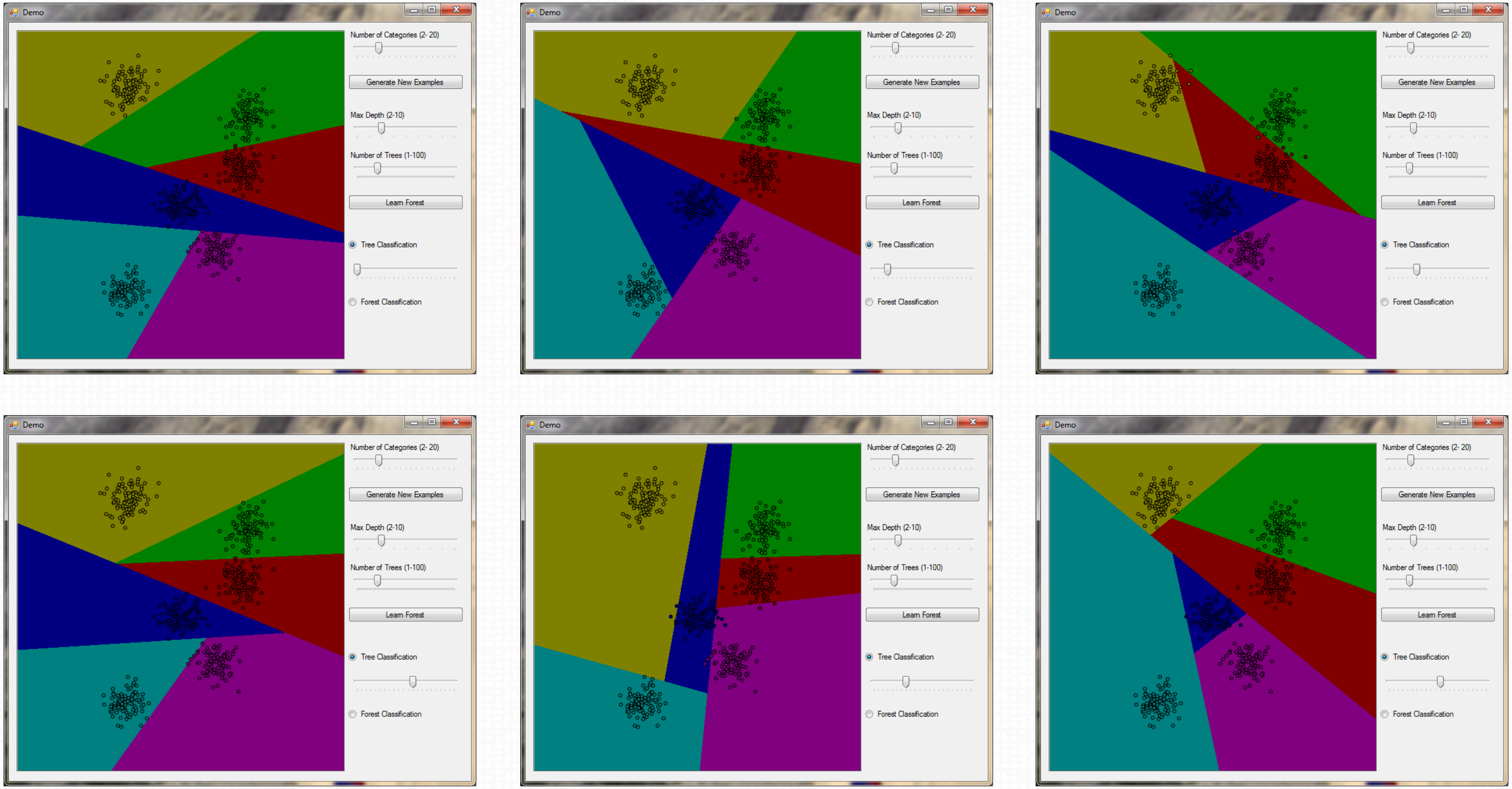
With a depth 3 tree, you are doing better, but still cannot separate all six classes.

Random forest: demo



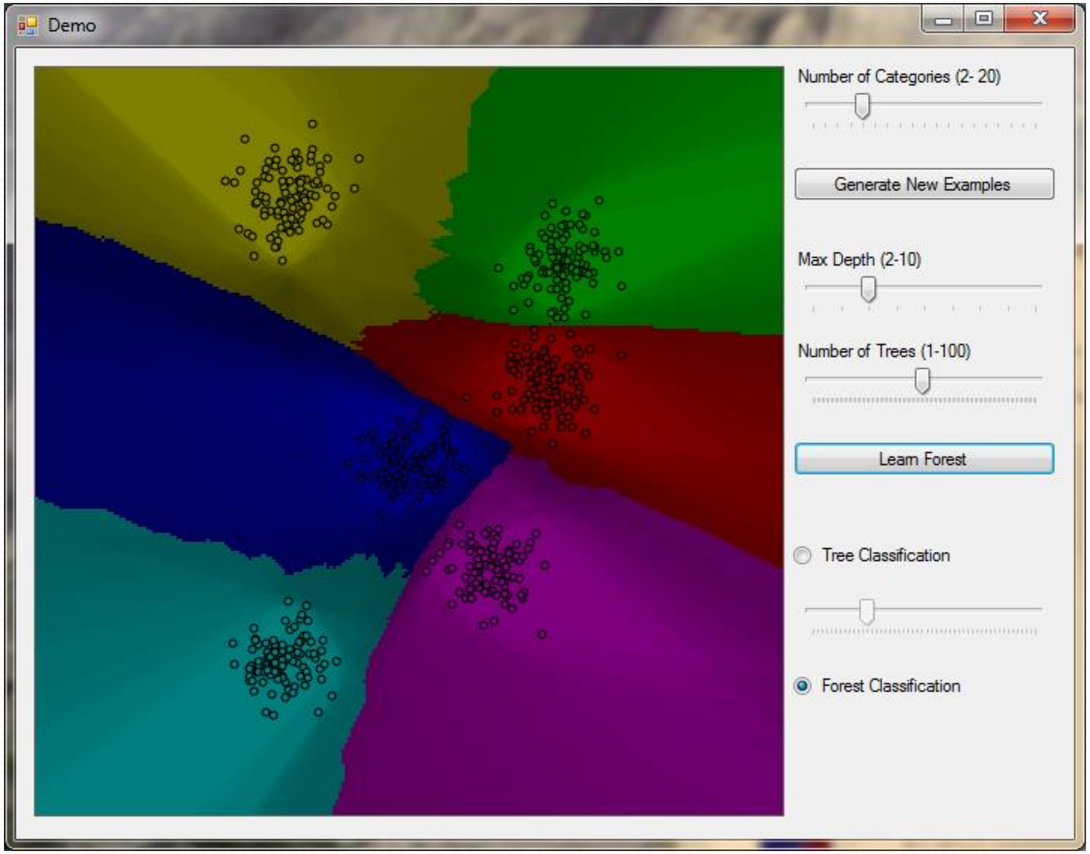
With a depth 4 tree, you now have at least as many leaf nodes as classes, and so are able to classify most examples correctly.

Random forest: demo



Different trees within a forest can give rise to very different decision boundaries, none of which is particularly good on its own.

Random forest: demo



But averaging together many trees in a forest can result in decision boundaries that look very sensible, and are even quite close to the max margin classifier.

Summary

- Very fast classification algorithm
- Accuracy comparable with other classifiers
- Simple to implement

Further reading/References

- ICCV'2009 Tutorial
  - http://jamie.shotton.org/work/presentations/ICCV2009TutorialPartI.pptx
- Random Forests for Regression and Classification; by Adele Cutler
  - http://www.math.usu.edu/adele/RandomForests/Ovronnaz.pdf
- Machine Learning; by Tom Mitchell (Chapter 3)
- Pattern Classification; By Duda, Hart, Stork (Chapter 8)

That's All

Thank You