Boosting & Randomized Forests for Visual Recognition

Jamie Shotton
Microsoft Research Cambridge

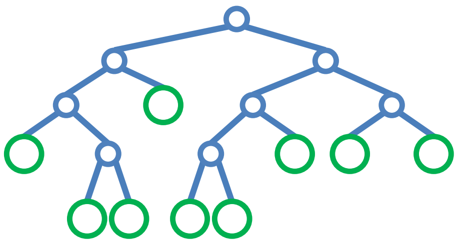
Tae-Kyun Kim
University of Cambridge

Björn Stenger
TOSHIBA

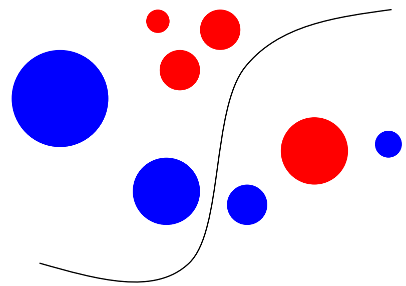
ICCV 2009, Kyoto, Japan

http://mi.eng.cam.ac.uk/~tkk22/iccv09_tutorial

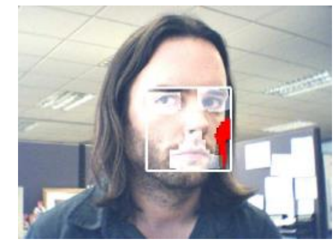
Course Overview



Part I:
Random Forests
Jamie Shotton



Part II:
Boosting
Tae-Kyun Kim



Part III:
Online Learning
Björn Stenger

- Coffee break
  - half way through Part II
- Questions
  - please ask as we go
- References & web resources
  - at end of each part
- Notation
  - may differ slightly between parts

http://mi.eng.cam.ac.uk/~tkk22/iccv09_tutorial

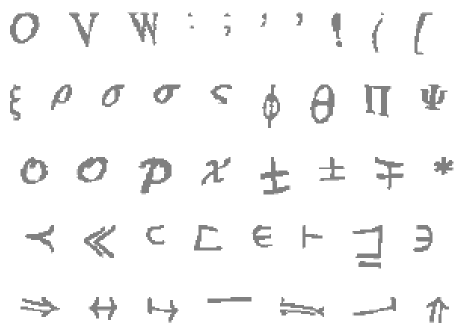
Part I

Randomized Decision Forests

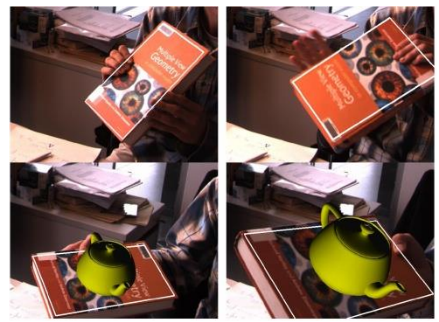
- Very fast tools for
  - classification
  - clustering
  - regression
- Good generalization through randomized training
- Inherently multi-class
  - automatic feature sharing [Torralba *et al.* 07]
- Simple training / testing algorithms

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

Randomized Forests in Vision

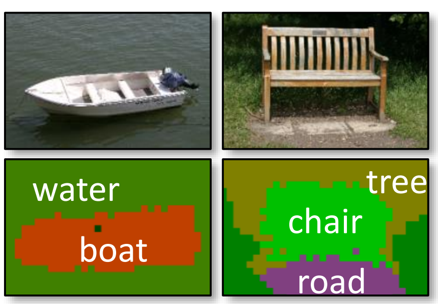


[Amit & Geman, 97]
digit recognition



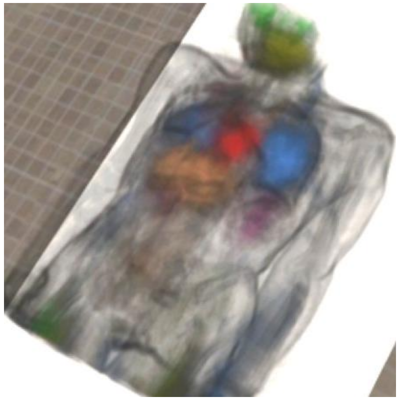
[Lepetit *et al.*, 06]
keypoint recognition

[Moosmann *et al.*, 06]
visual word clustering



[Shotton *et al.*, 08]
object segmentation

[Rogez *et al.*, 08]
pose estimation



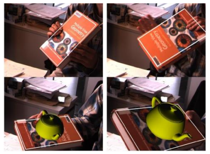
[Criminisi *et al.*, 09]
organ detection

(Among many others...)

Outline

- **Randomized Forests**
  - motivation
  - training & testing
  - implementation
  - regression, clustering, max-margin, boosting

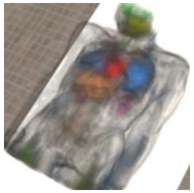
- **Applications to Vision**
  - keypoint recognition
  - object segmentation
  - human pose estimation
  - organ detection



[Lepetit *et al.*, 06]
keypoint recognition

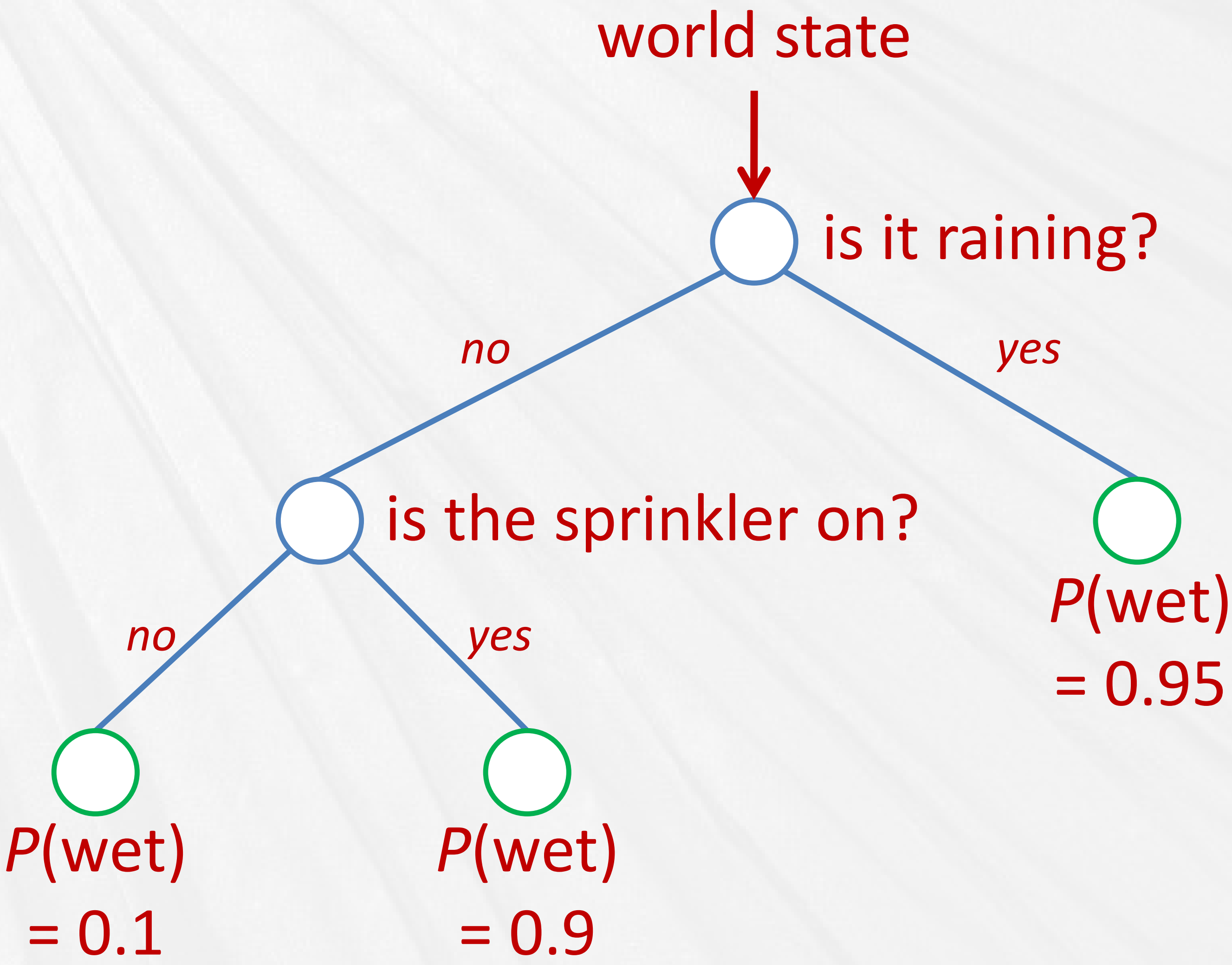
[Shotton *et al.*, 08]
object segmentation

[Rogez *et al.*, 08]
pose estimation



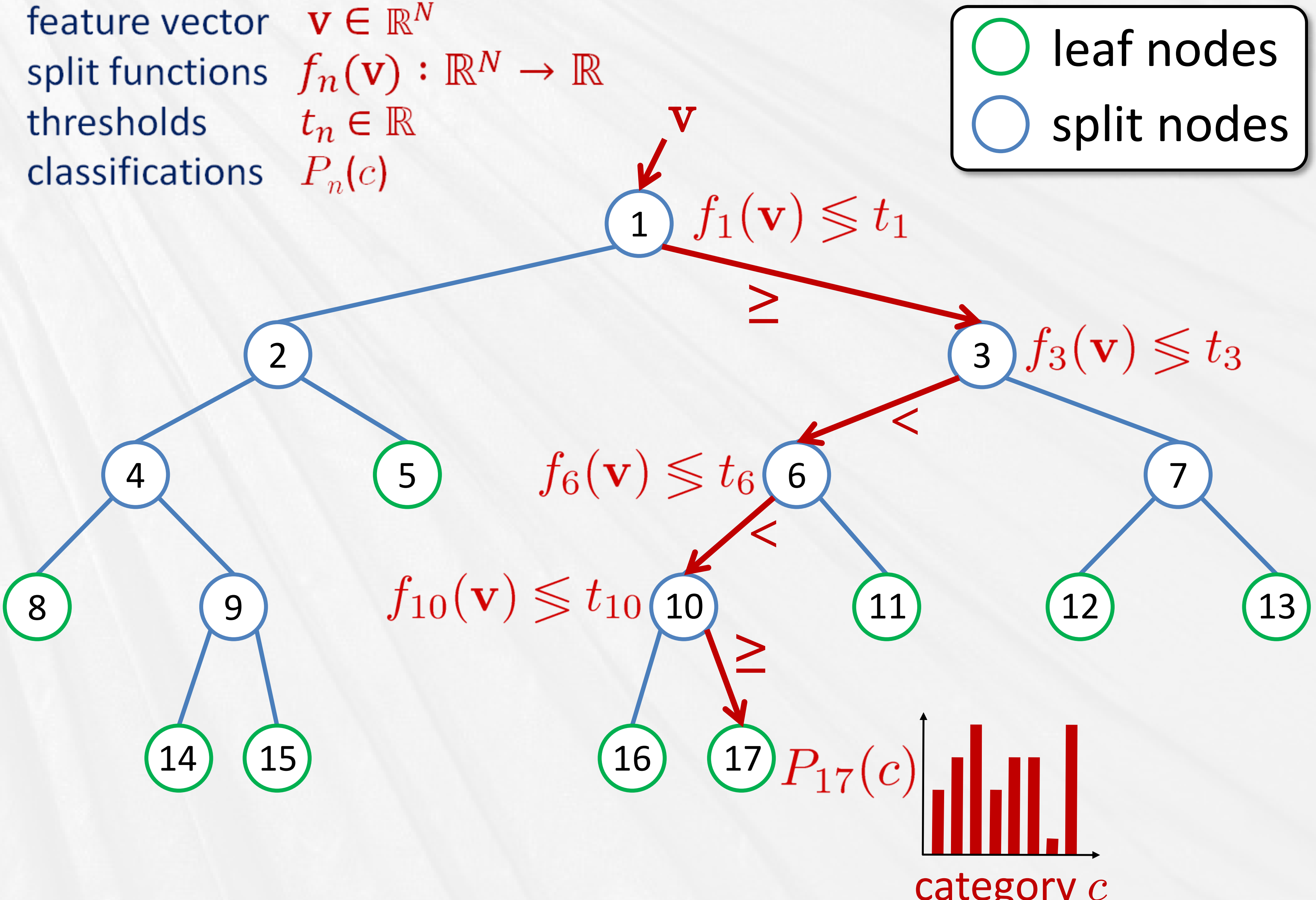
[Criminisi *et al.*, 09]
organ detection

The Basics: **Is The Grass Wet?**



The Basics: **Binary Decision Trees**

- feature vector $\mathbf{v} \in \mathbb{R}^N$
- split functions $f_n(\mathbf{v}) : \mathbb{R}^N \rightarrow \mathbb{R}$
- thresholds $t_n \in \mathbb{R}$
- classifications $P_n(c)$

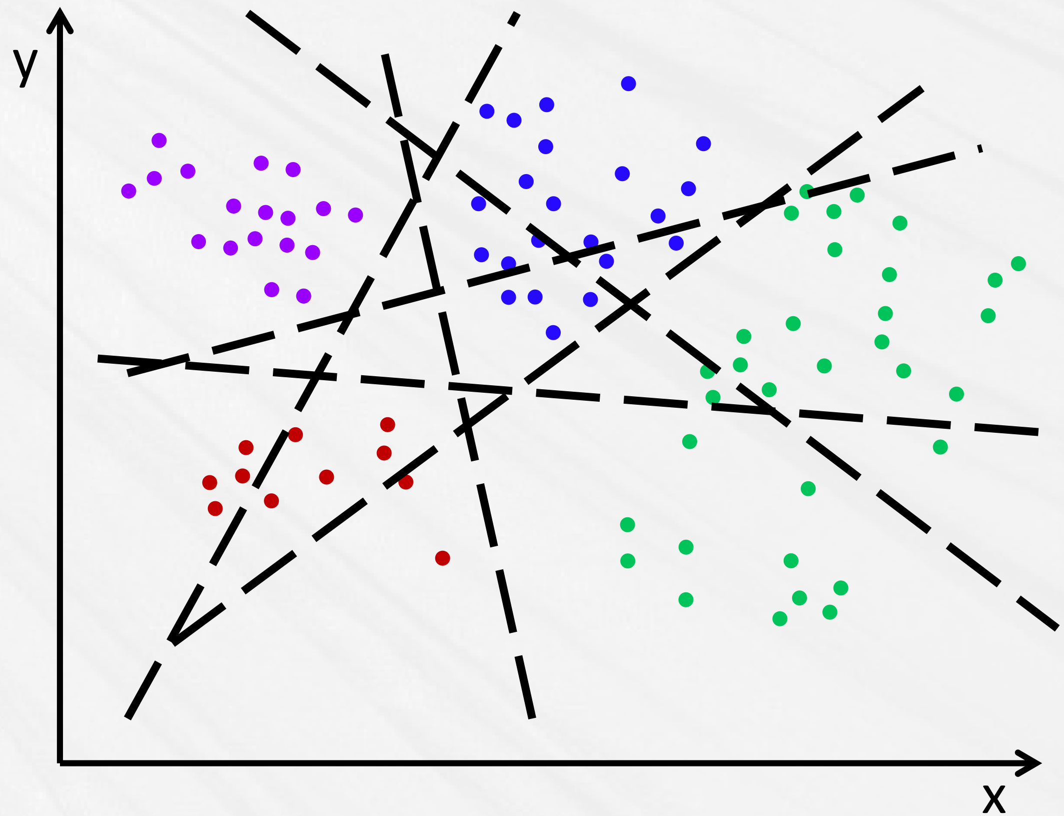


Decision Tree **Pseudo-Code**

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

Toy Learning Example

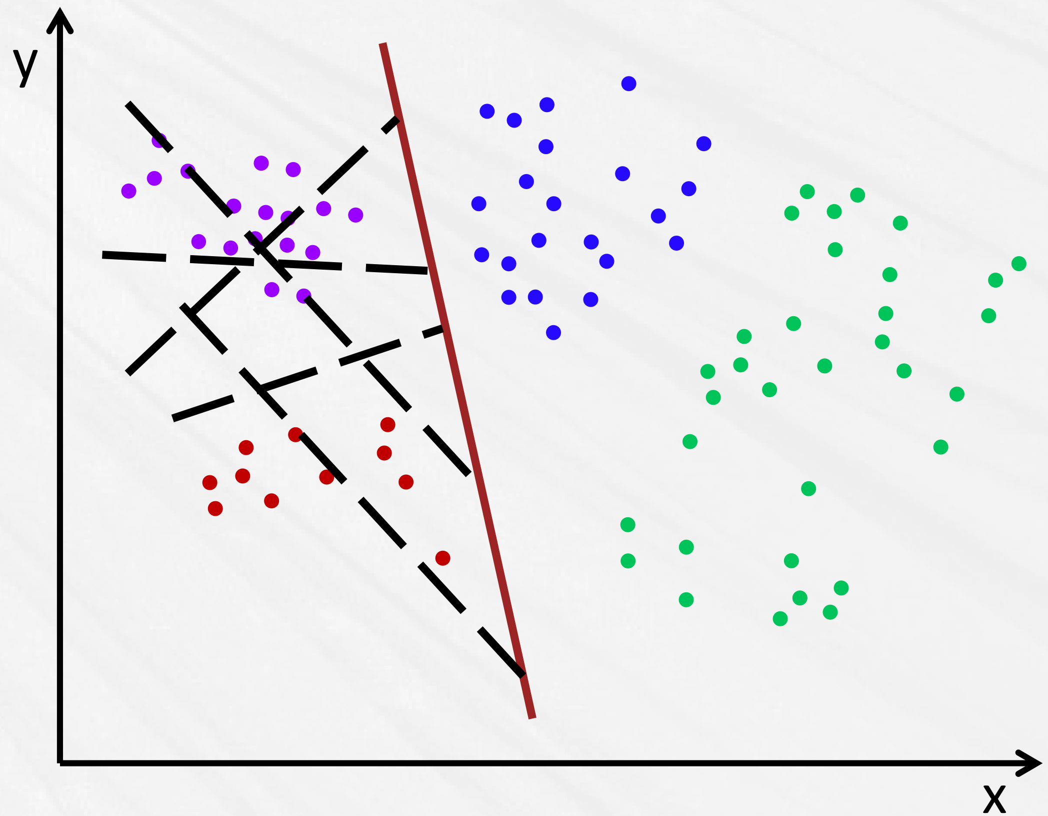
- **Try several lines, chosen at random**
- **Keep line that best separates data**
  - information gain
- **Recurse**



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

Toy Learning Example

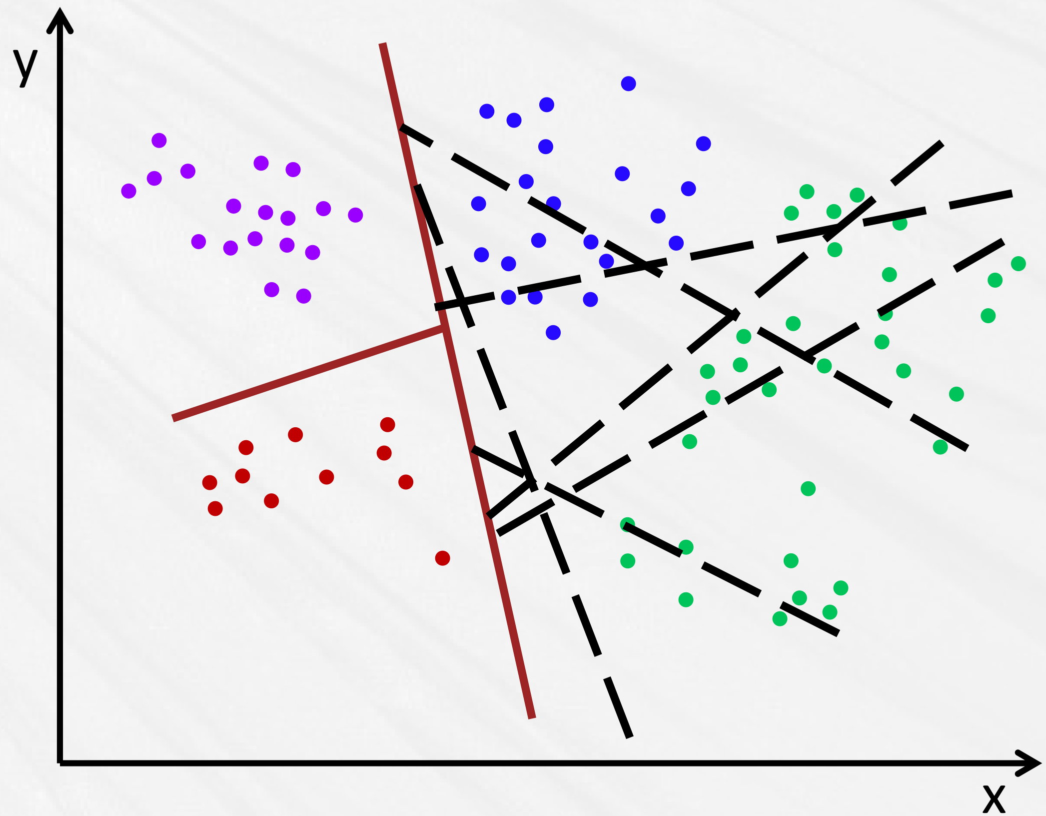
- **Try several lines, chosen at random**
- **Keep line that best separates data**
  - information gain
- **Recurse**



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

Toy Learning Example

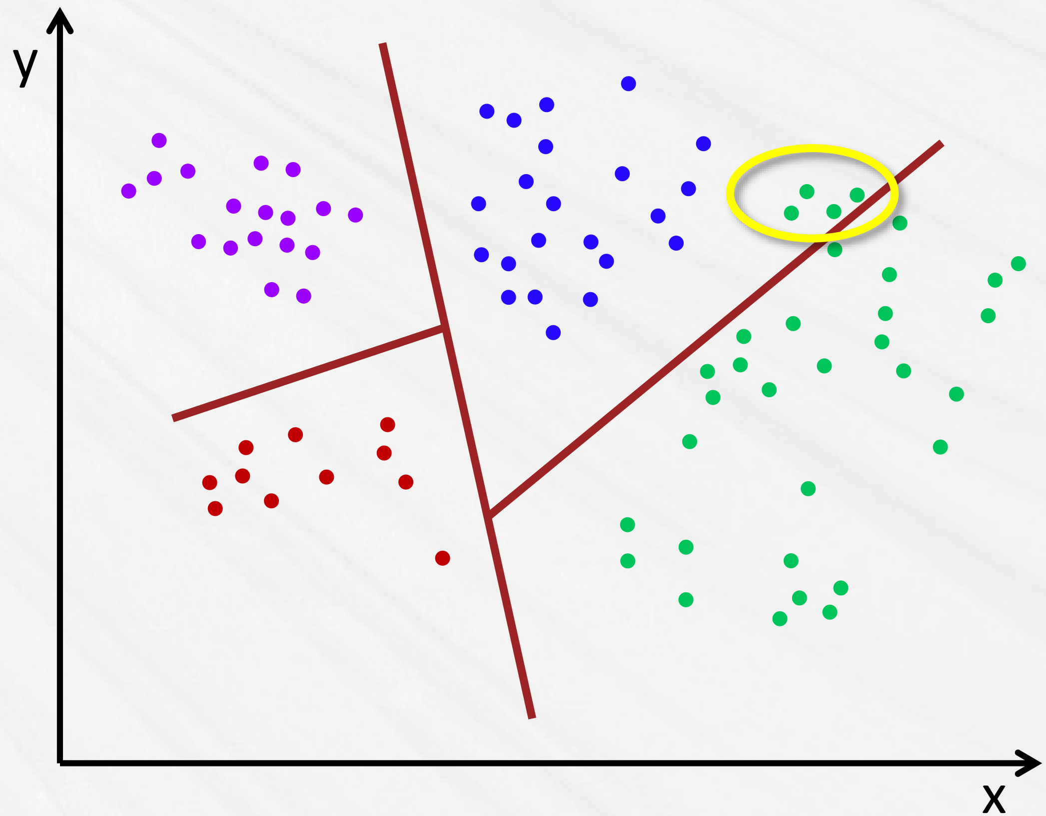
- **Try several lines, chosen at random**
- **Keep line that best separates data**
  - information gain
- **Recurse**



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

Toy Learning Example

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



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

Randomized Learning

- **Recursively split examples at node n**
  - set I_n indexes labeled training examples ($\mathbf{v}_i$, l_i):

left split

$$I_\mathrm{l} = \{i \in I_n \mid f(\mathbf{v}_i) < t\}$$

$$I_\mathrm{r} = I_n \setminus I_\mathrm{l}$$

right split

threshold

function of example i's feature vector

- **At node n, $P_n(c)$ is histogram of example labels l_i**

More **Randomized Learning**

$$\begin{array}{lcl} \text{left split} \quad I_{\mathrm{l}} & = & \{i \in I_n \mid f(\mathbf{v}_i) < t\} \\ \text{right split} \quad I_{\mathrm{r}} & = & I_n \setminus I_{\mathrm{l}} \end{array}$$

- **Features** $f(\mathbf{v})$ **chosen at random from feature pool** $f \in F$
- **Thresholds** t **chosen in range** $t \in (\min_i f(\mathbf{v}_i), \max_i f(\mathbf{v}_i))$
- **Choose** f **and** t **to maximize gain in information**

$$\Delta E = -\frac{|I_{\mathrm{l}}|}{|I_n|}E(I_{\mathrm{l}}) - \frac{|I_{\mathrm{r}}|}{|I_n|}E(I_{\mathrm{r}})$$

Entropy E calculated from histogram of labels in I

Implementation Details

- **How many features and thresholds to try?**
  - just one = "extremely randomized" [Geurts *et al.* 06]
  - few -> fast training, may under-fit, maybe too deep
  - many -> slower training, may over-fit

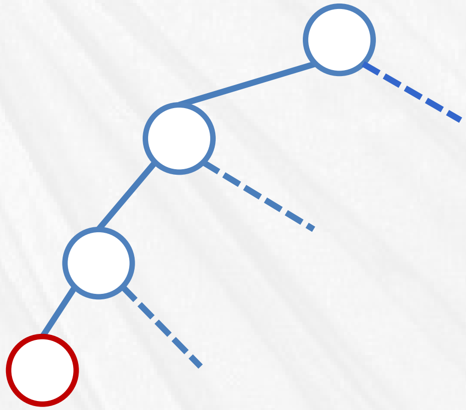
- **When to stop growing the tree?**
  - maximum depth
  - minimum entropy gain
  - delta class distribution
  - pruning

Randomized Learning **Pseudo Code**

```
TreeNode LearnDT(I)

   repeat featureTests times
      let f = RndFeature()
      let r = EvaluateFeatureResponses(I, f)

      repeat threshTests times
         let t = RndThreshold(r)
         let (I_l, I_r) = Split(I, r, t)
         let gain = InfoGain(I_l, I_r)
         if gain is best then remember f, t, I_l, I_r
      end
   end

   if best gain is sufficient
      return SplitNode(f, t, LearnDT(I_l), LearnDT(I_r))
   else
      return LeafNode(HistogramExamples(I))
   end
end
```

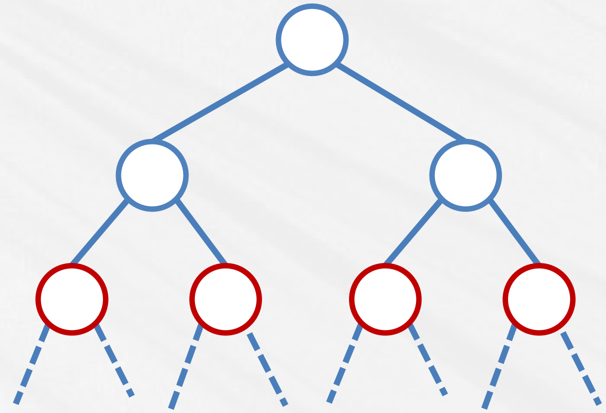
Training Strategies

depth first



- **Recursive algorithm**
  - partitions all training examples
- **Store all images in memory**
  - can be memory hungry
- **Good for**
  - smaller data sets
  - deeper trees

breadth first



- **One pass through data per tree level**
  - can load images on-the-fly
- **Maintain 4D histogram of size**

$$2^d \times F \times T \times C$$

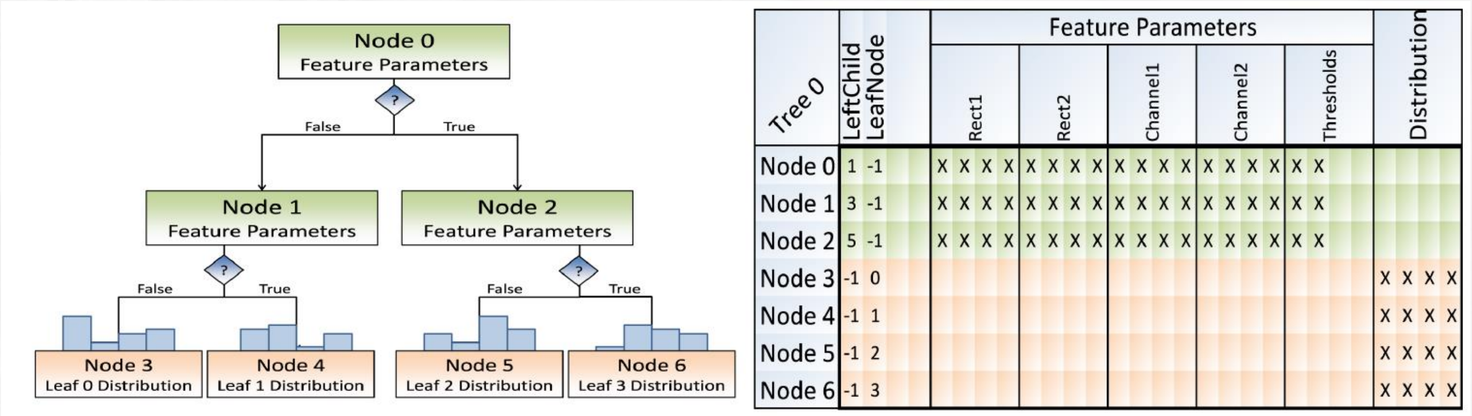
no. nodes, no. features, no. thresholds, no. classes

- **Good for**
  - very large data sets
  - shallower trees

GPU Acceleration [Sharp 08]

- **GPUs can dramatically accelerate**
  - training – 10x speed-up – breadth first
  - testing – 100x speed-up

- **Tree is encoded as GPU texture**



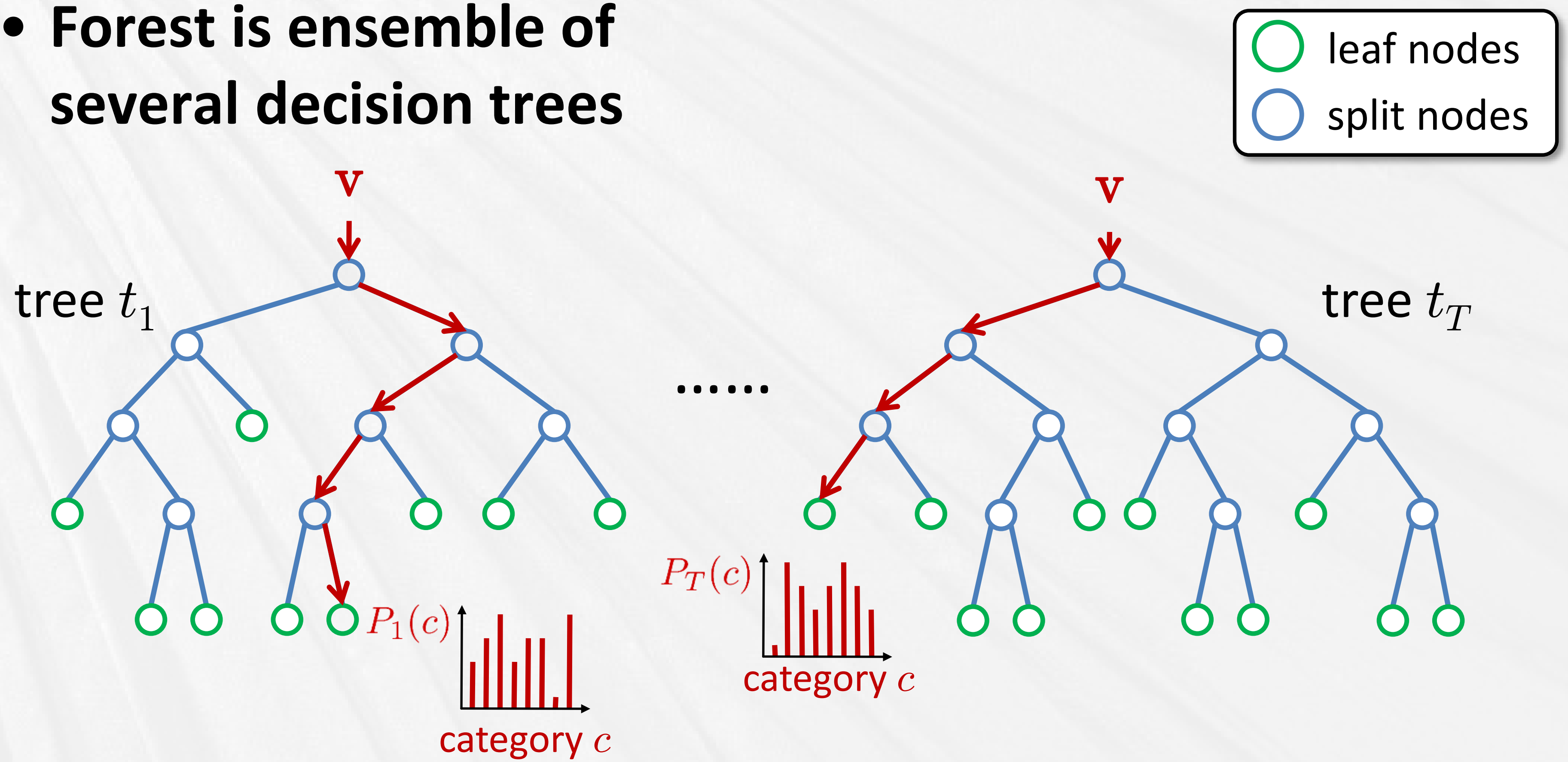
- **Caveats**
  - some limitations on image features
  - implementation requires considerable GPU know-how

Binary Decision Trees Summary

- **Fast greedy training algorithms**
  - 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
- **Prone to over-fitting**

A Forest of Trees

- **Forest is ensemble of several decision trees**



– classification is $P(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|\mathbf{v})$

[Amit & Geman 97]
[Breiman 01]
[Lepetit *et al.* 06]

Decision Forests **Pseudo-Code**

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

    // loop over trees in forest
    for t = 1 to forest.CountTrees
        let P’ = ClassifyDT(forest.Tree[t], v)
        P = P + P’ // sum distributions
    end

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

Learning a Forest

- **Divide training examples into T subsets $I_t \subseteq I$**
  - **improves generalization**
  - reduces **memory requirements** & **training time**

- **Train each decision tree t on subset I_t**
  - same decision tree learning as before

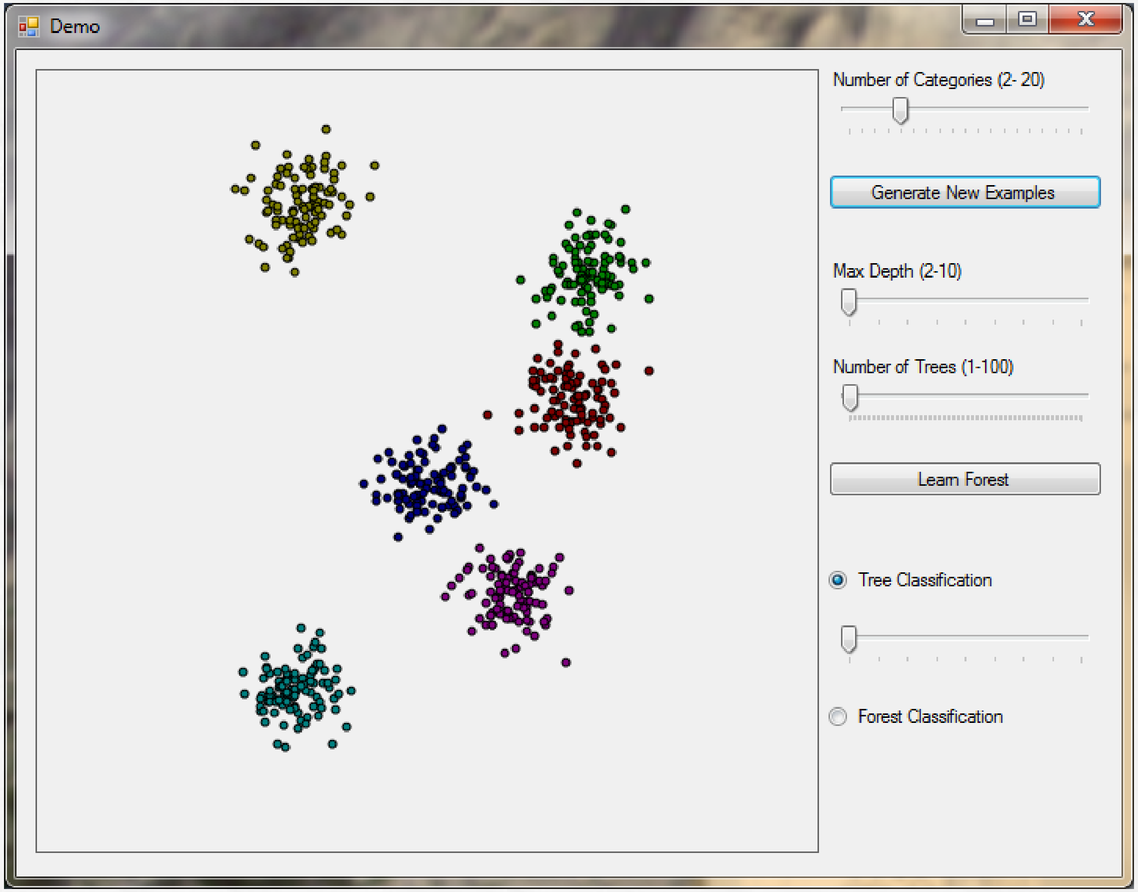
- **Multi-core friendly**

> - Subsets can be chosen at random or hand-picked
> - Subsets can have overlap (and usually do)
> - Can enforce subsets of *images* (not just examples)
> - Could also divide the feature pool into subsets

Learning a Forest **Pseudo Code**

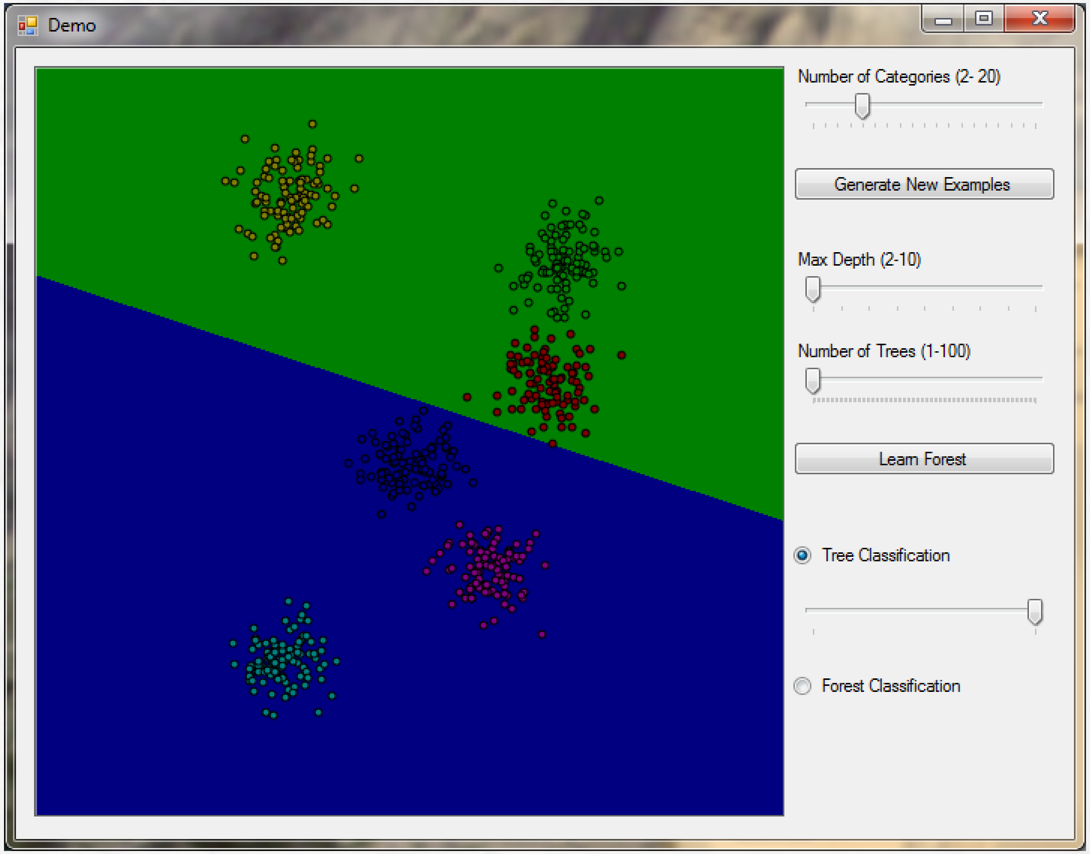
```
Forest LearnDF(countTrees, I)
    // allocate memory
    let forest = Forest(countTrees)

    // loop over trees in forest
    for t = 1 to countTrees
        let I_t = RandomSplit(I)
        forest[t] = LearnDT(I_t)
    end

    // return forest object
    return forest
end
```

Toy Forest Classification Demo



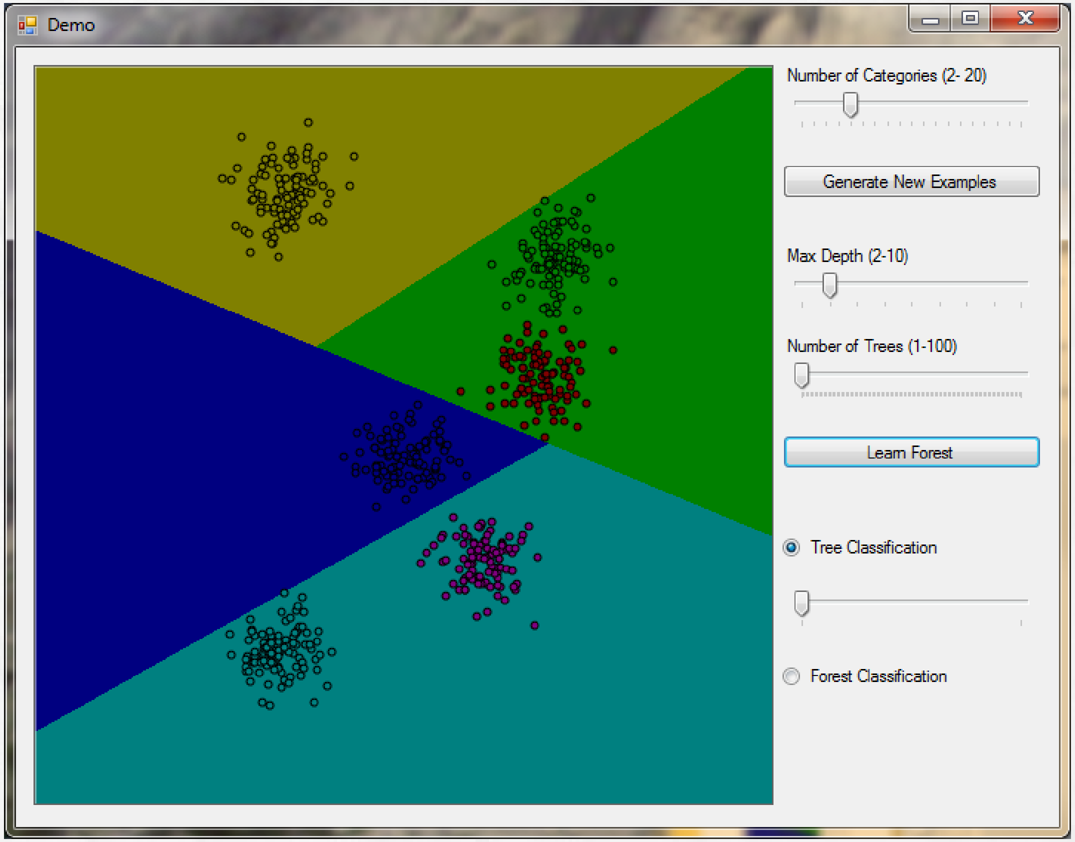
6 classes in a 2 dimensional feature space.
Split functions are lines in this space.

Toy Forest Classification Demo



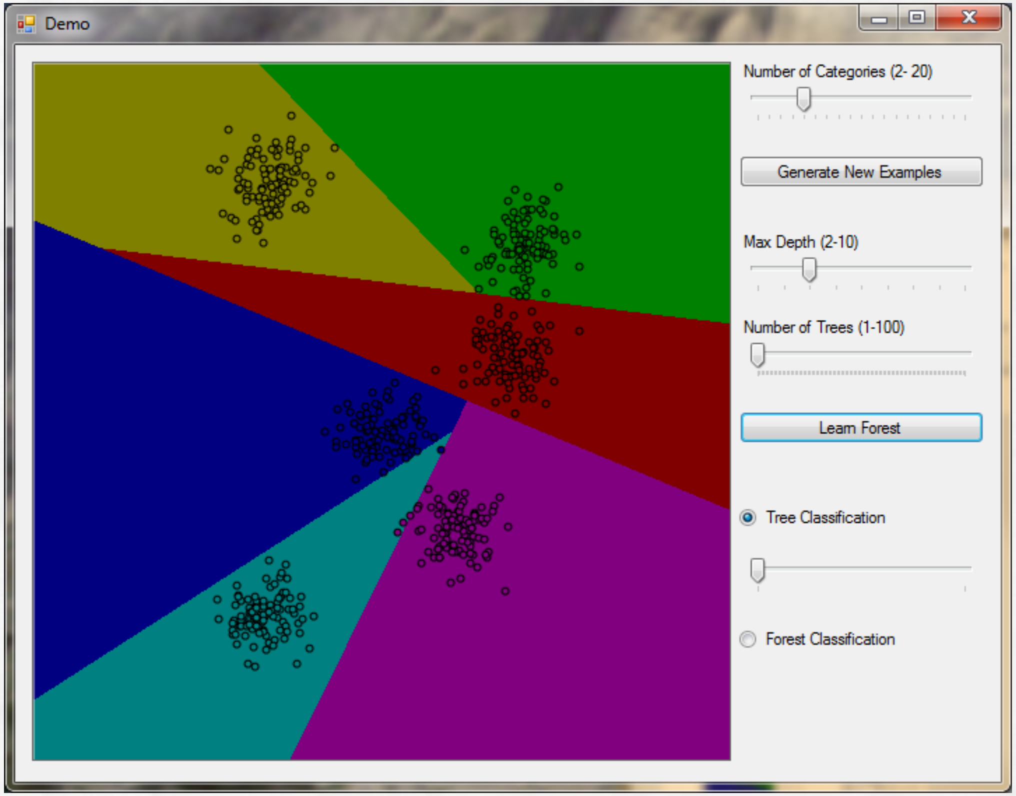
With a depth 2 tree, you cannot separate all six classes.

Toy Forest Classification Demo



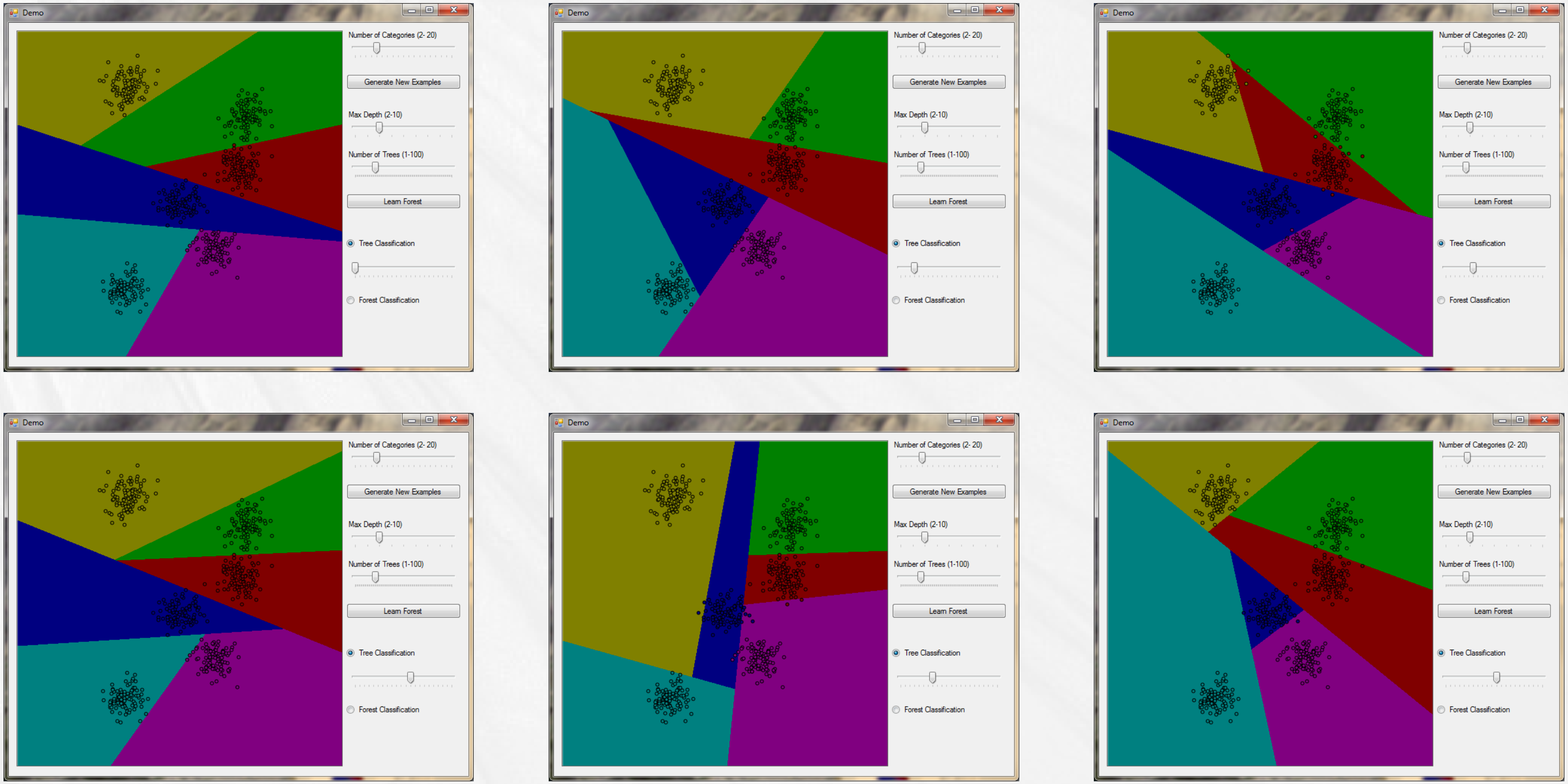
With a depth 3 tree, you are doing better, but still cannot separate all six classes.

Toy Forest Classification Demo



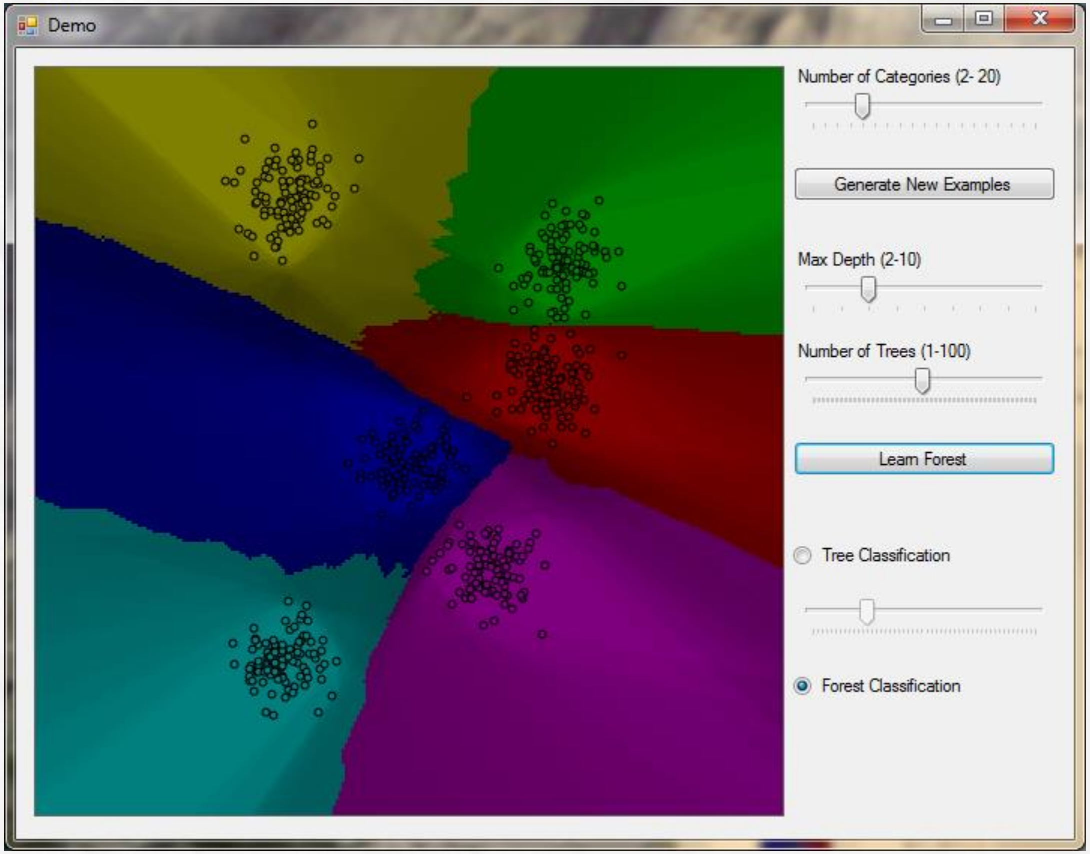
With a depth 4 tree, you now have at least as many leaf nodes as classes, and so are able to classify most examples correctly.

Toy Forest Classification Demo



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

Toy Forest Classification Demo

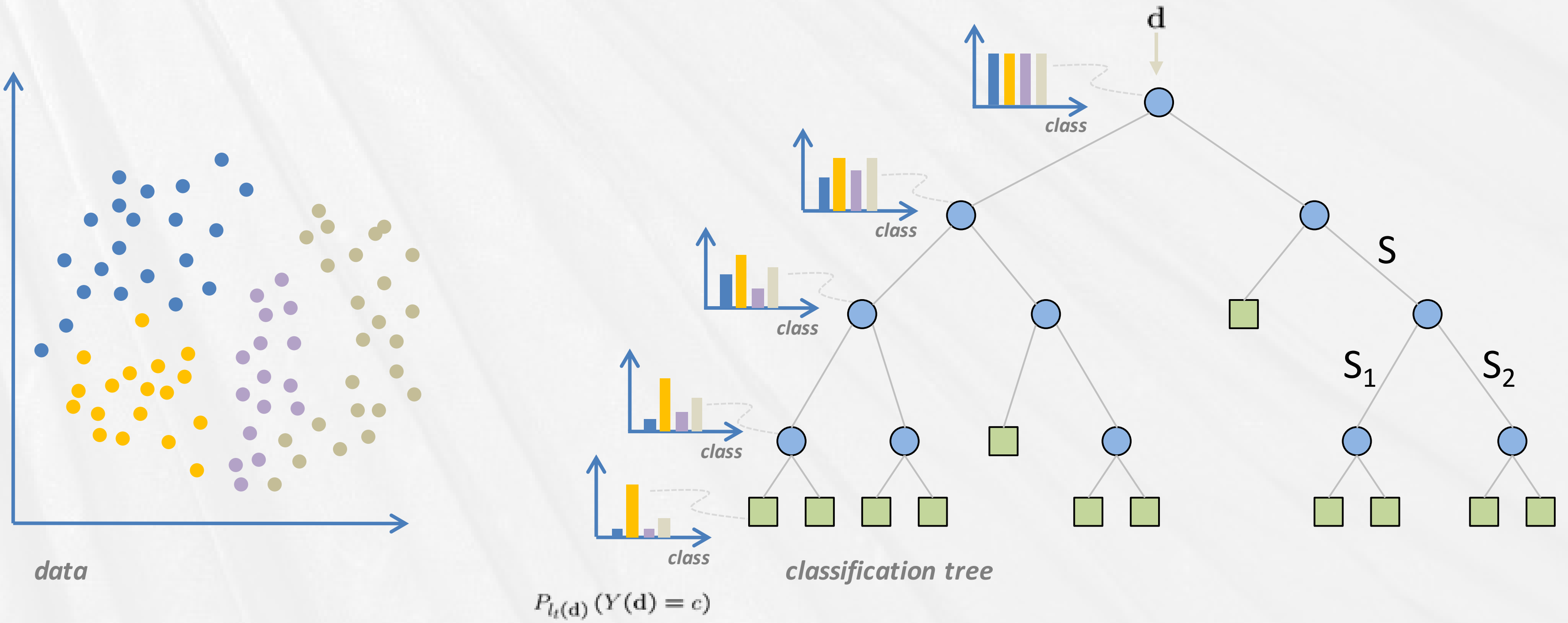


But averaging together many trees in a forest can result in decision boundaries that look very sensible, and are even quite close to the max margin classifier. (Shading represents entropy – darker is higher entropy).

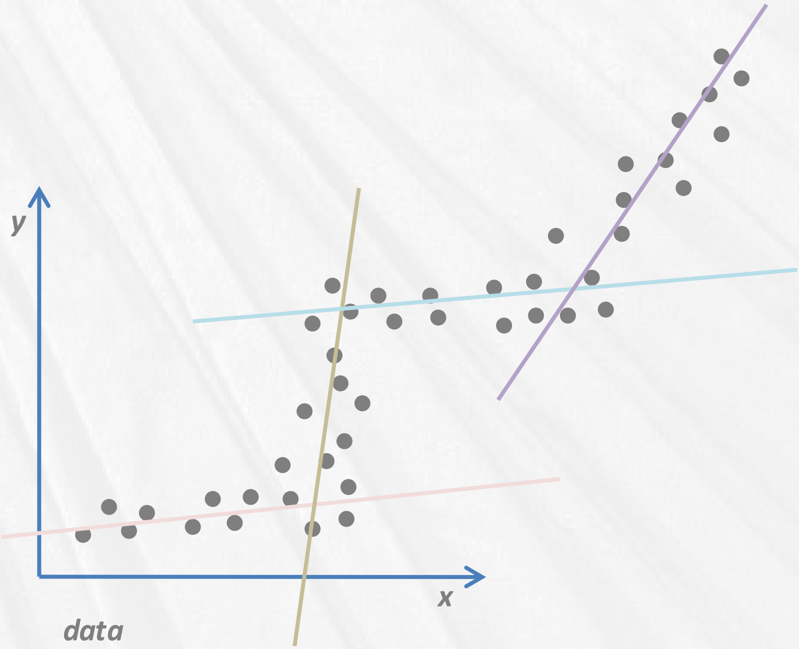
Tree outputs and objective functions

- **Trees can be trained for**
  - classification, regression, or clustering

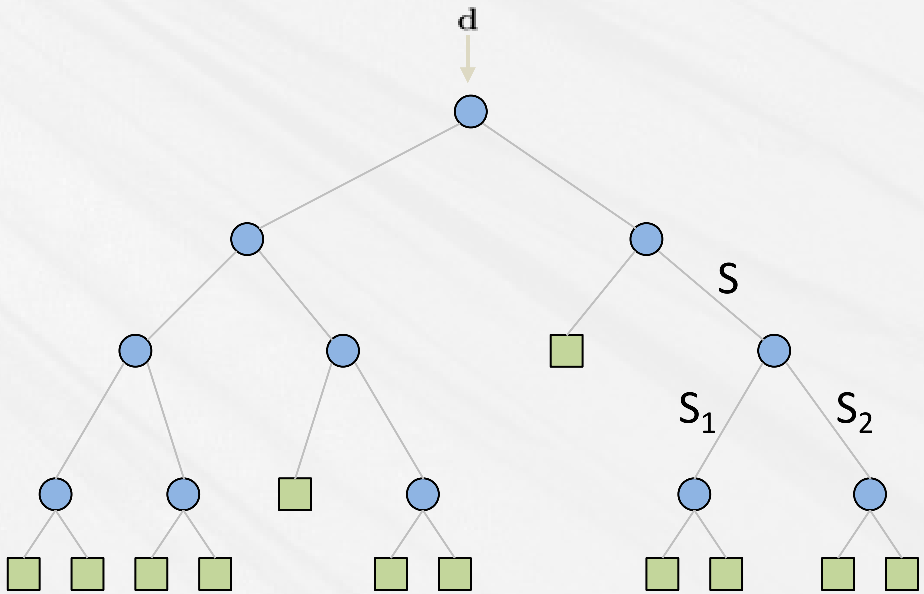
- **Change the object function**
  - information gain for classification: $I = H(S) - \sum_{i=1}^{2} \frac{|S_i|}{|S|} H(S_i)$ measure of distribution purity



Regression trees



data



regression tree

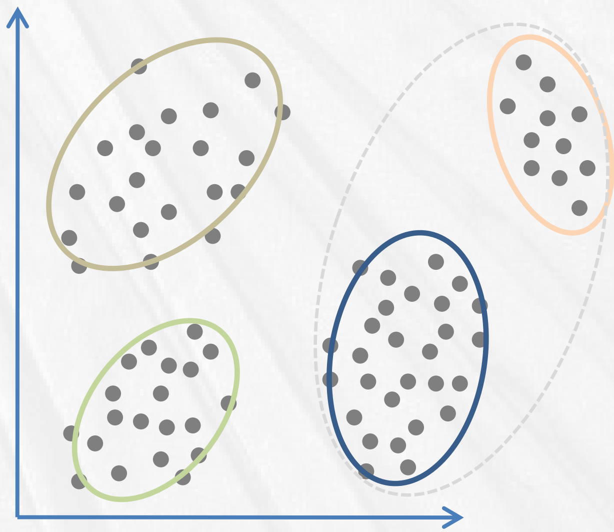
- Real-valued output *y*
- Object function: maximize $Err(S) - \sum_{i=1}^{2} \frac{|S_i|}{|S|} Err(S_i)$

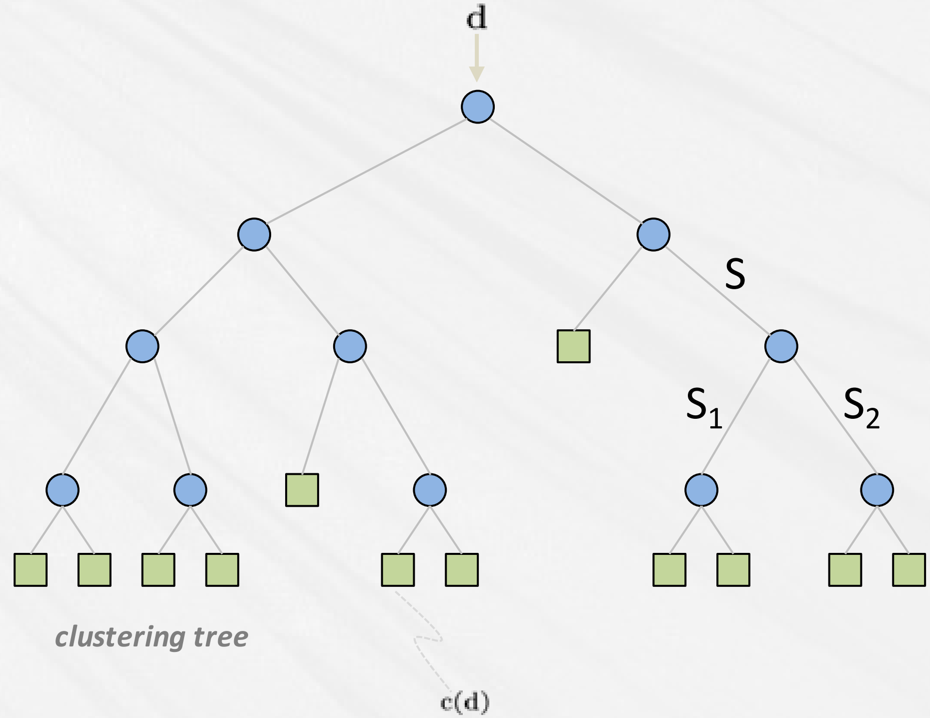
measure of fit of model

$$Err(S) = \sum_{j \in S} \left(y_j - y(x_j) \right)^2$$

e.g. linear model y = ax+b,
Or just constant model

Clustering trees





- Output is cluster membership
- Option 1 – minimize imbalance: $B = |\log|S_1| - \log|S_2||$ [Moosmann *et al.* 06]
- Option 2 – maximize Gaussian likelihood:

$$T = |\Lambda_S| - \sum_{i=1}^{2} \frac{|S_i|}{|S|} |\Lambda_{S_i}|$$

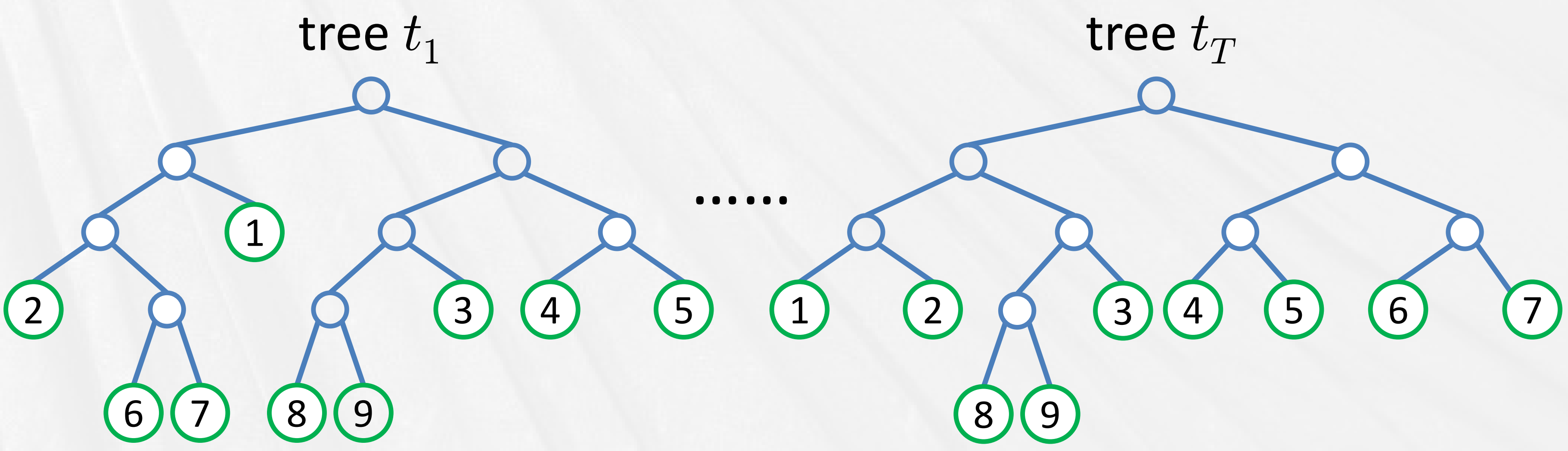
measure of cluster tightness
(maximizing a function of info gain
for Gaussian distributions)

Clustering example

[Moosmann *et al.* 06]

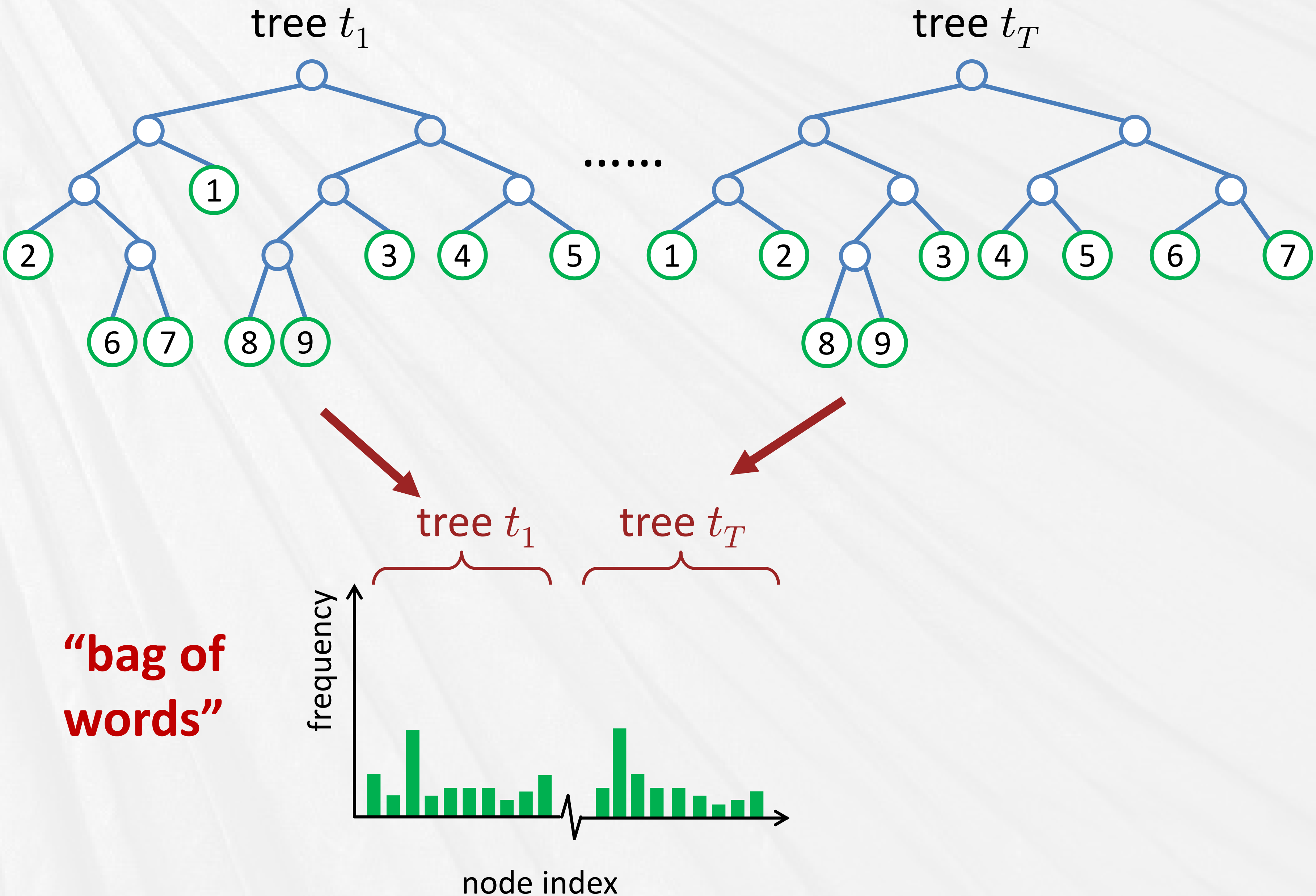
[Sivic *et al.* 03]
[Csurka *et al.* 04]

- **Visual words good for e.g. matching, recognition but *k*-means clustering very slow**
- **Randomized forests for clustering descriptors**
  - e.g. SIFT, texton filter-banks, etc.
- **Leaf nodes in forest are clusters**
  - concatenate histograms from trees in forest



Clustering example

[Moosmann *et al.* 06]



"bag of words"

Relation to other parts of this tutorial

- **Boosting (Part II)**
  - decision trees as weak learners
  - boosted classifiers as split functions [Tu 05]

- **Online learning (Part III)**
  - trees can be updated 'online' [Yeh *et al.* 07]
    - distributions of leaves
    - structure of tree

Relation to Cascades

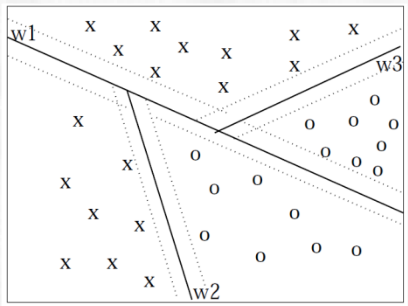
[Viola & Jones 04]

- **Boosted Cascades**
  - very unbalanced tree
  - good for unbalanced binary problems
    e.g. sliding window object detection

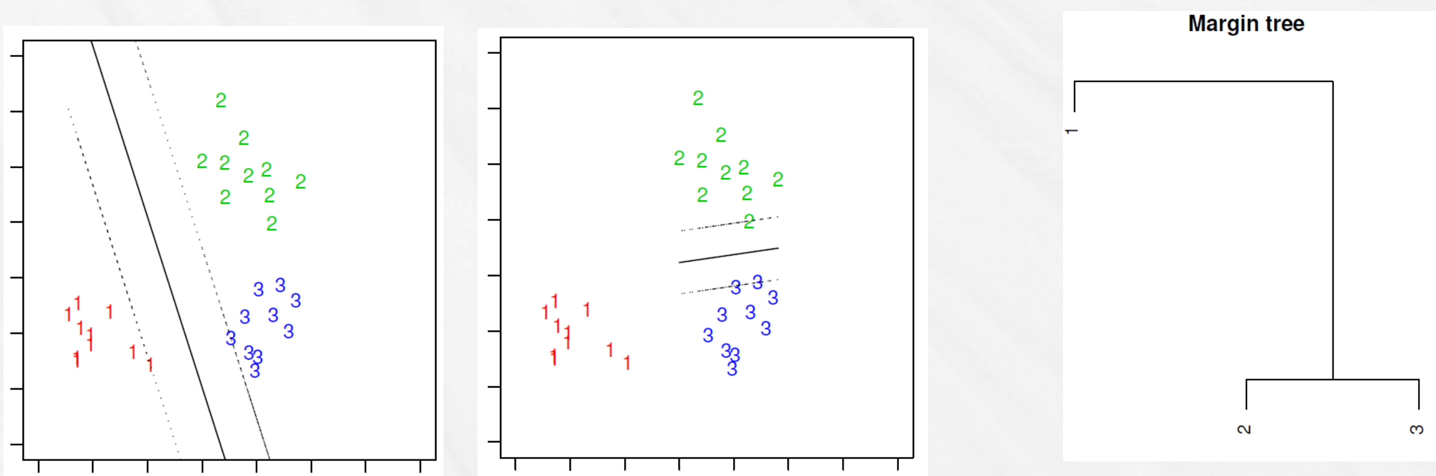
- **Randomized forests**
  - less deep, fairly balanced
  - ensemble of trees gives robustness
  - good for multi-class problems

Relation to Max-Margin Classifiers

- **Max-margin split functions** [Wu *et al.*, 00]
  - split functions have built-in generalization



- **Tree of max-margin classifiers (SVMs)** [Tibshirani & Hastie, 07]
  - recursively partition set of classes down the tree



Random Ferns

- **Naïve Bayes classifier over random sets of features**

$$P(C|f_1, ... f_N) \quad \propto \quad P(f_1, ... f_N|C)P(C)$$ **Bayes' rule**

$$\approx \quad \prod_{j=1}^{N} P(\underbrace{f_j}_{\text{individual features}}|C)$$ **"naïve Bayes"**

$$\approx \quad \prod_{k=1}^{M} P(\underbrace{F_k}_{\text{set of features}}|C)$$ **"random ferns"**

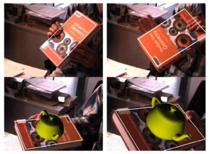
- **Can be good alternative to randomized forests**

[Özuysal *et al.* 07]
[Bosch *et al.* 07]

Outline

- **Randomized Forests**
  - motivation
  - training & testing
  - implementation
  - regression, clustering, max-margin, boosting

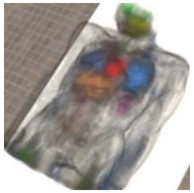
- **Applications to Vision**
  - keypoint recognition
  - object segmentation
  - human pose estimation
  - organ detection



[Lepetit *et al.*, 06]
keypoint recognition

[Shotton *et al.*, 08]
object segmentation

[Rogez *et al.*, 08]
pose estimation

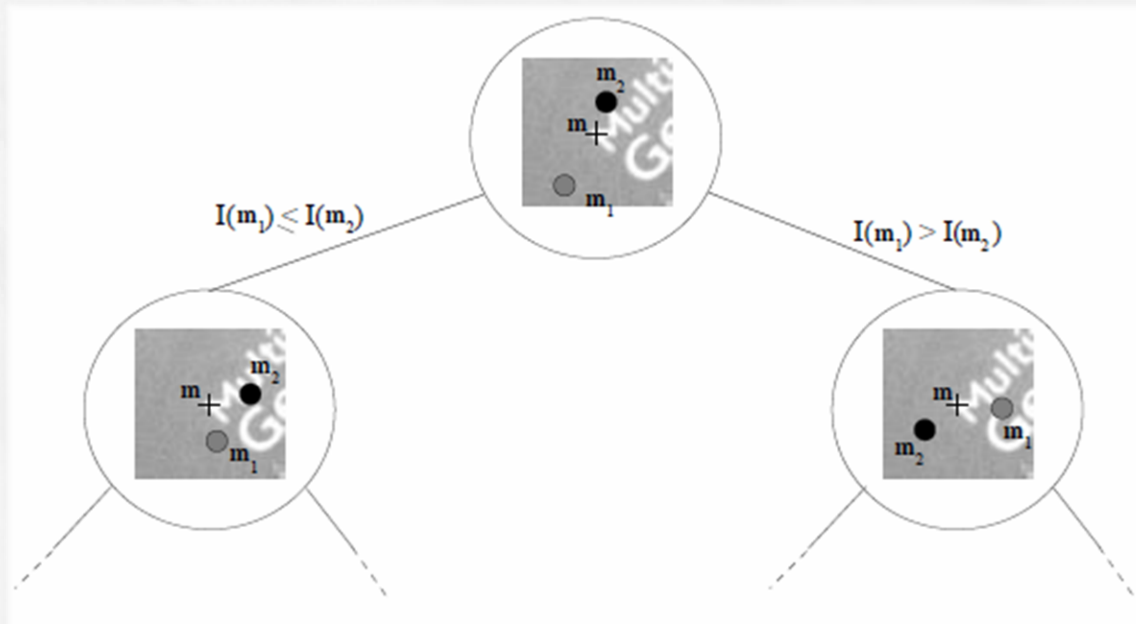


[Criminisi *et al.*, 09]
organ detection

Fast Keypoint Recognition [Lepetit *et al.* 06]

- **Wide-baseline matching as classification problem**

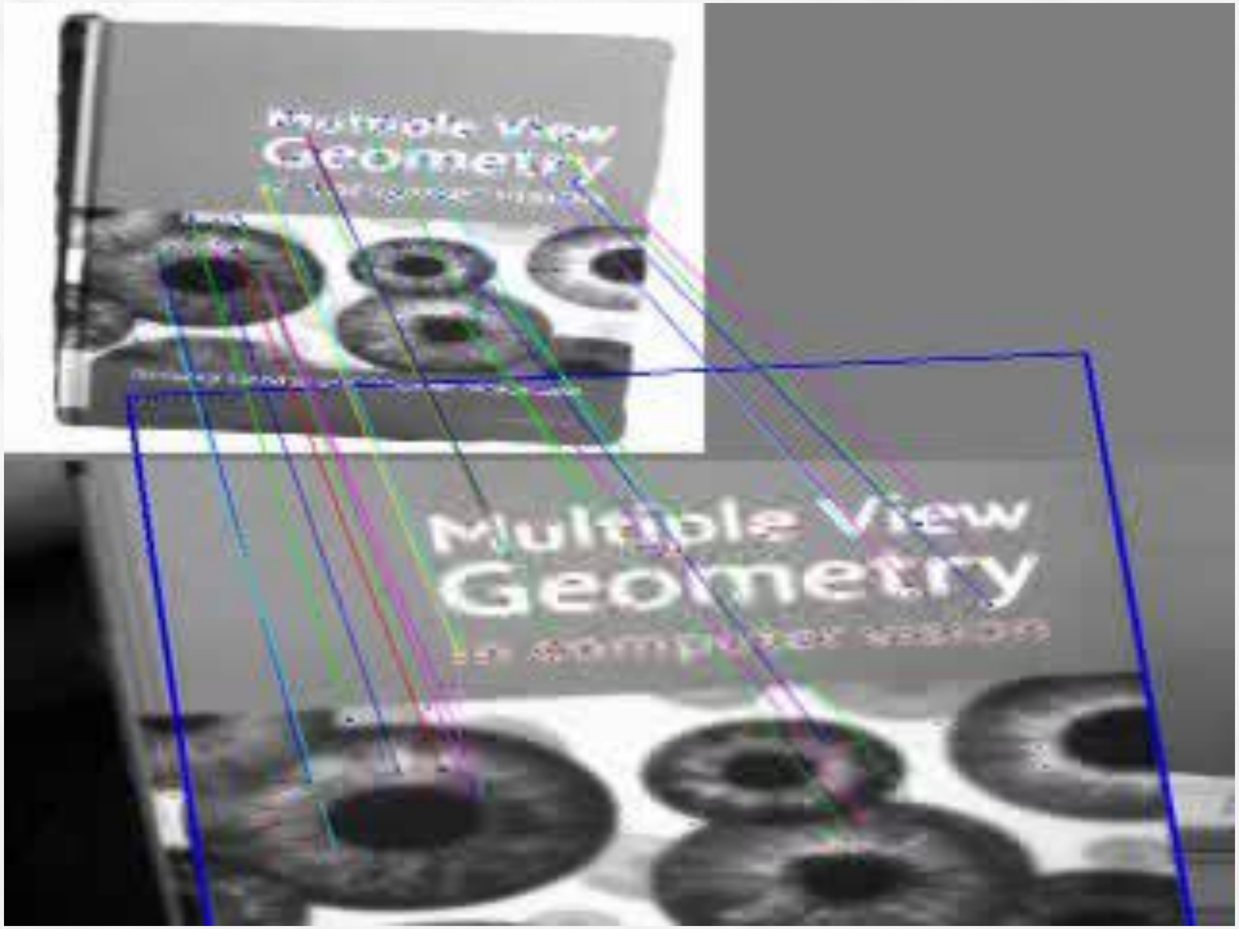
- **Extract prominent key-points in training images**



- **Forest classifies**
  - patches -> keypoints
- **Features**
  - pixel comparisons
- **Augmented training set**
  - gives robustness to patch scaling, translation, rotation

Fast Keypoint Recognition [Lepetit *et al.* 06]

- **Example videos**
  - from http://cvlab.epfl.ch/research/augm/detect.php



Real-Time Object Segmentation [Shotton *et al.* 2008]

- **Segment image and label segments in real-time**

CVPR 2008 Best Demo Award!

Object Recognition Pipeline

extract features

SIFT, filter bank

clustering

k-means

assignment

nearest neighbour

hand-crafted

unsupervised

classification algorithm

SVM, decision forest, boosting

supervised

Object Recognition Pipeline

classification algorithm

SVM, decision forest, boosting

Semantic Texton Forest **(STF)**

- decision forest for **clustering** & **classification**
- tree nodes have learned object category associations

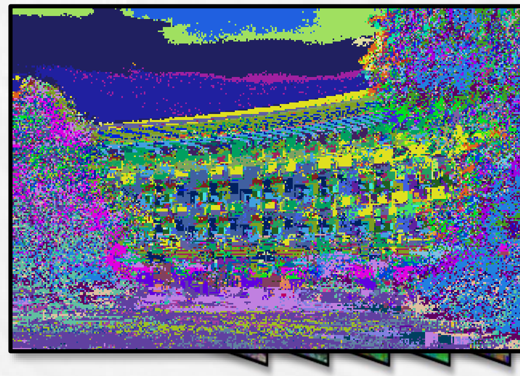
Example Semantic Texton Forest

Leaf Node Visualization

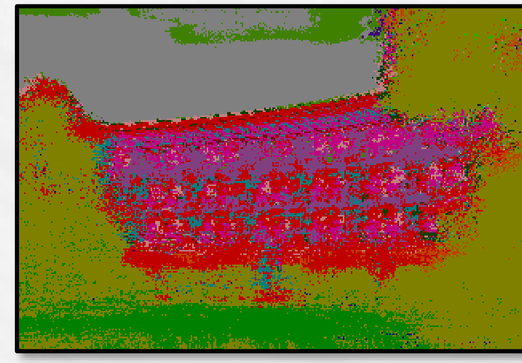
- **Average of all training patches at each leaf node**

Semantic Textons & Local Classification

test image



semantic textons
(color ⇔ leaf node index)



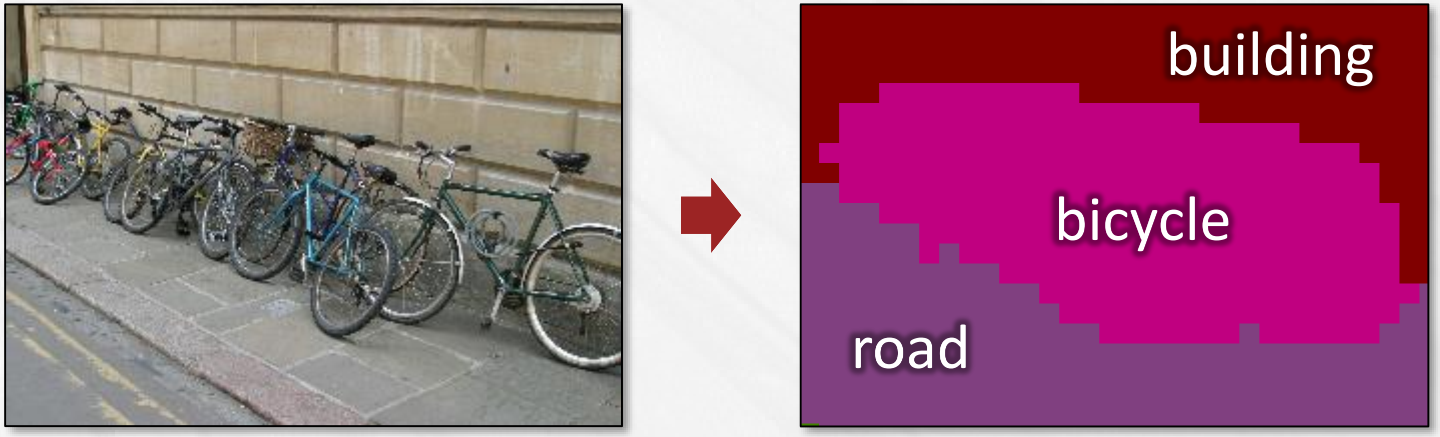
local classification
(color ⇔ most likely category)

ground truth
(for reference)

comparable

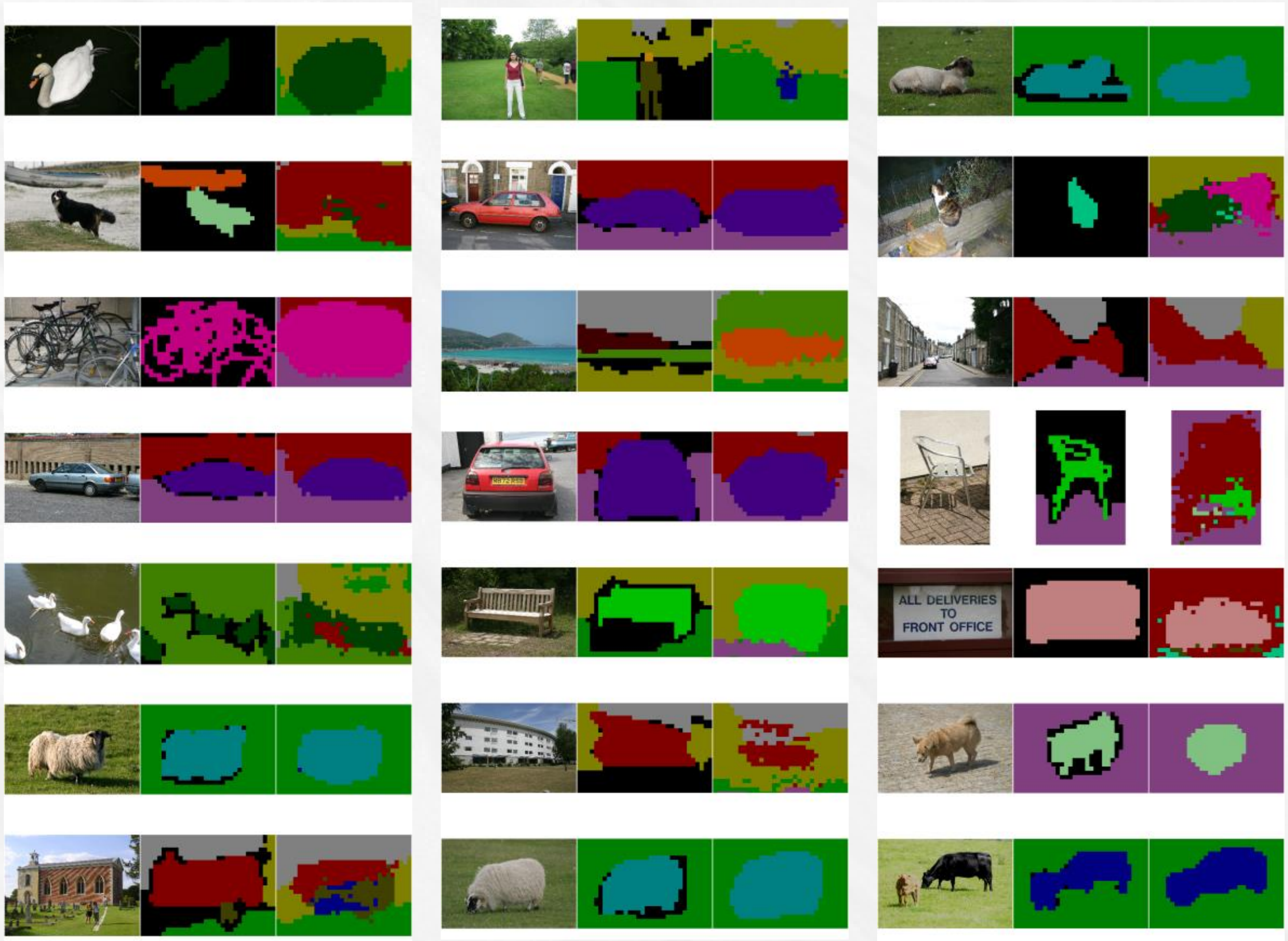
Segmentation Forest

- **Object segmentation**



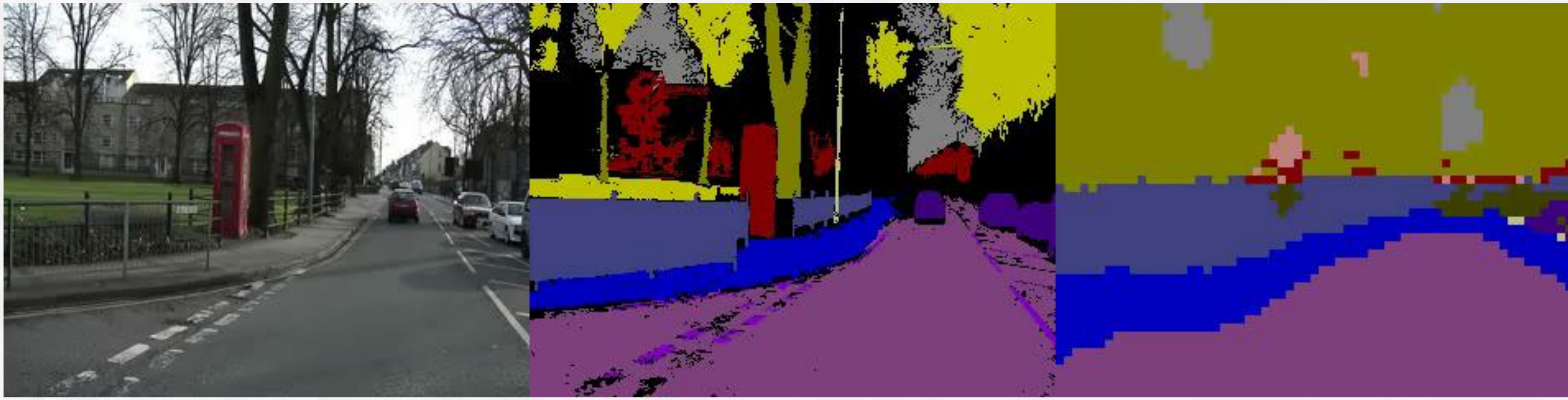
- **Adapt TextonBoost [Shotton *et al.* 07]**
  - boosted classifier → randomized decision forest
    textons → semantic textons

MSRC Dataset Results



building	grass	tree	cow	sheep	sky	airplane	water	face	car	boat
bicycle	flower	sign	bird	book	chair	road	cat	dog	body	

3D Point-Cloud Features

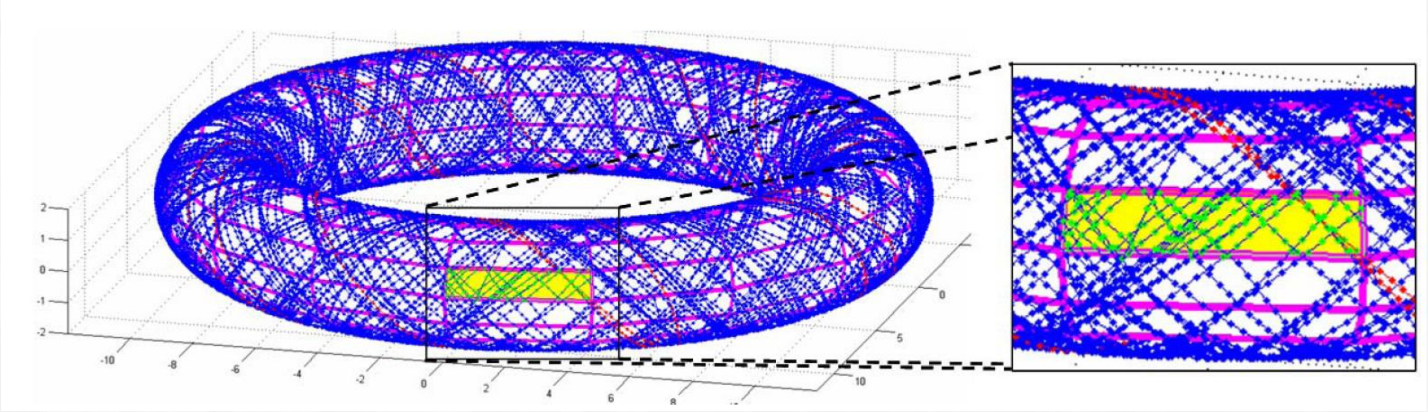


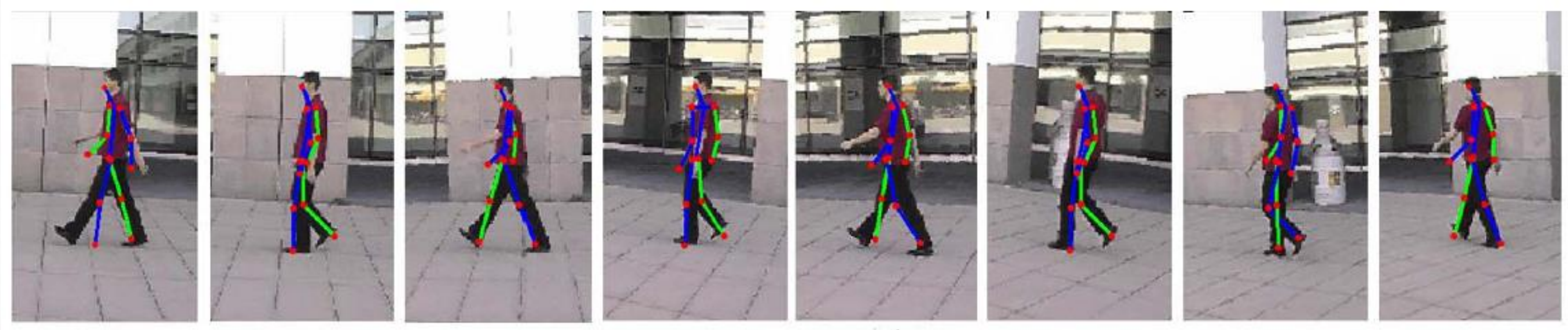
test image ground truth result

- **[Brostow *et al.* 08]**
  – structure-from-motion cues for object segmentation

Human Pose Estimation [Rogez *et al.* 08]

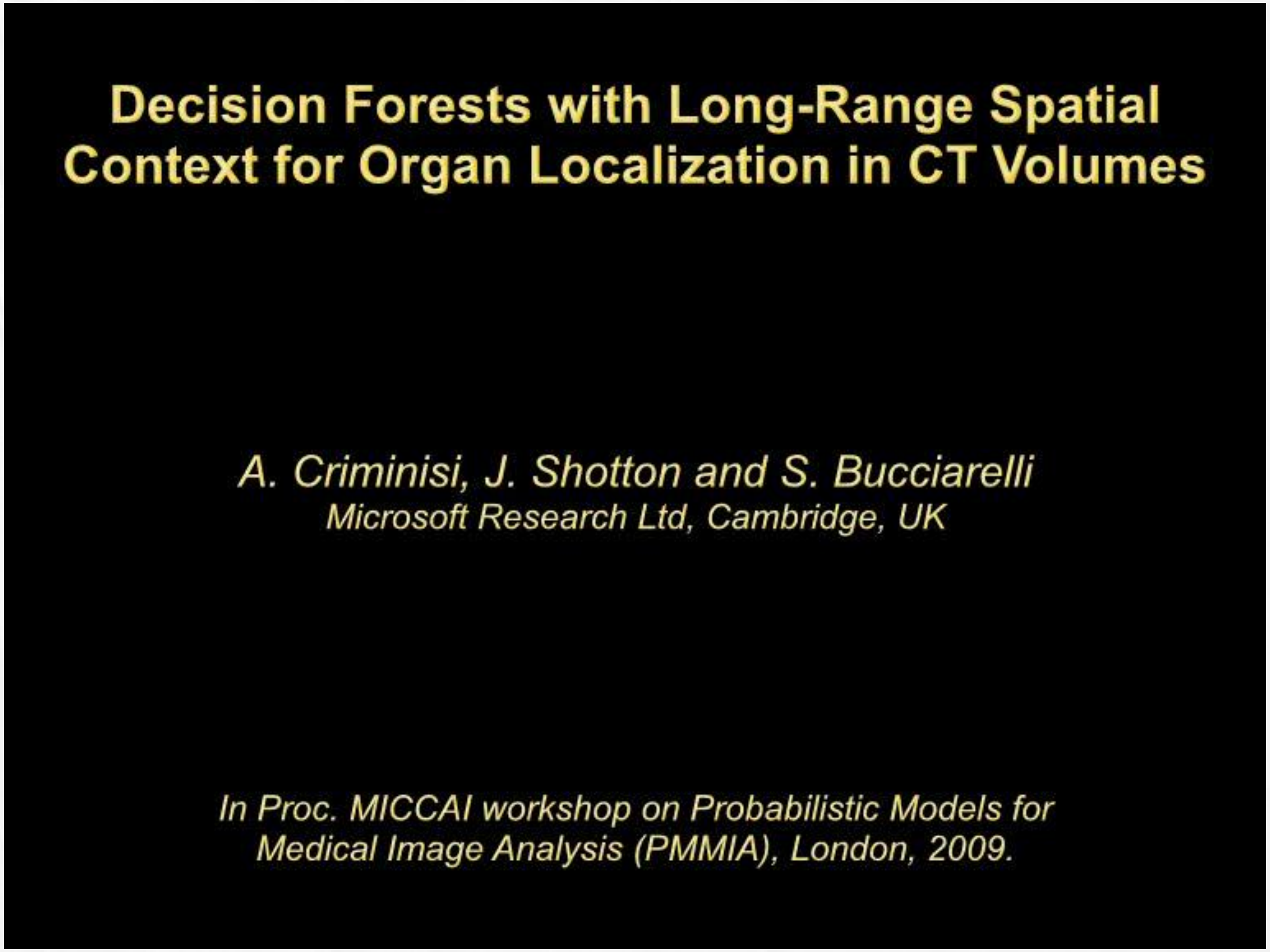
- **Torus defined on**
  - dimension 1: cyclical action (e.g. walking)
  - dimension 2: camera view point (360 degrees)
- **Discrete bins on the torus used as classes in random forest**





Organ Recognition [Criminisi *et al.* 09]

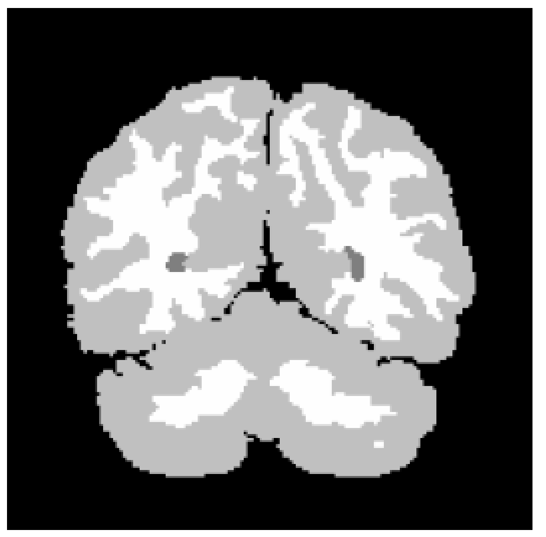
- **Quickly localize bodily organs in 3D CT scans**

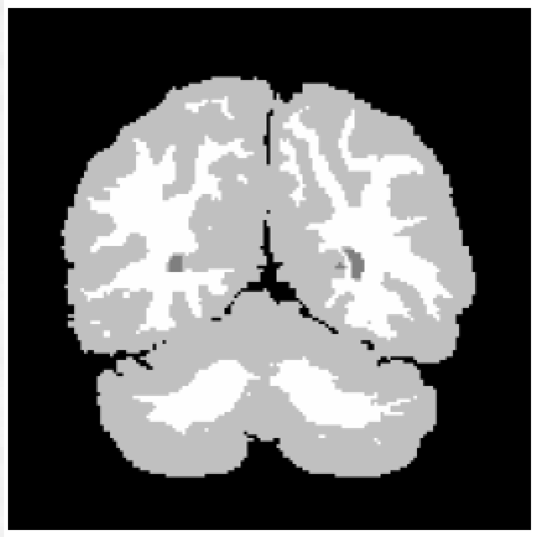


Brain Segmentation

[Yi *et al.* MICCAI 09]

ground truth





result

Method	CSF	GM	WM
Adaptive MAP	0.069	0.564	0.567
Biased MAP	0.071	0.558	0.562
Fuzzy c-means	0.048	0.473	0.567
Maximum-a-posteriori (MAP)	0.071	0.550	0.554
Maximum-likelihood	0.062	0.535	0.551
Tree-Structure k-means	0.049	0.477	0.571
MPM-MAP [11]	0.227	0.662	0.683
MAP with histograms	0.549 ± 0.017	0.814 ± 0.004	0.710 ± 0.005
Decision Forest Classifier	$\mathbf{0.614 \pm 0.015}$	$\mathbf{0.838 \pm 0.006}$	$\mathbf{0.731 \pm 0.007}$

Take Home Message from Part I

- **Randomized decision forests**
  - very fast
  - accuracy comparable with other classifiers
  - simple to implement
  - extremely flexible tools for computer vision

References (red = most relevant)

- **Amit & Geman**
  - Shape Quantization and Recognition with Randomized Trees.
  - Neural Computation 1997.
- **Bosch *et al.***
  - Image Classification using Random Forests and Ferns.
  - ICCV 2007.
- **Breiman**
  - Random Forests.
  - Machine Learning Journal 2001.
- **Breiman *et al.***
  - Classification and Regression Trees
  - Chapman & Hall, 1984.
- **Brostow *et al.***
  - Segmentation and Recognition using Structure from Motion Point Clouds.
  - ECCV 2008.
- **Csurka *et al.***
  - Visual Categorization with Bags of Keypoints.
  - ECCV Workshop on Statistical Learning in Computer Vision, 2004.
- **Fuchs & Buhmann**
  - Inter-Active Learning of Randomized Tree Ensembles for Object Detection.
  - ICCV Workshop on On-line Learning for Computer Vision, 2009.
- **Geurts *et al.***
  - Extremely Randomized Trees.
  - Machine Learning 2006.
- **Grauman & Darrel**
  - The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features.
  - ICCV 2005.
- **Hua *et al.***
  - Discriminant Embedding for Local Image Descriptors.
  - ICCV 2007.
- **Jurie & Triggs**
  - Creating Efficient Codebooks for Visual Recognition.
  - ICCV 2005.
- **Lazebnik *et al.***
  - Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories.
  - CVPR 2006.
- **Lepetit *et al.***
  - Keypoint Recognition using Randomized Trees.
  - PAMI 2006.
- **Lowe**
  - Distinctive image features from scale-invariant keypoints.
  - IJCV 2004.
- **Malik *et al.***
  - Contour and Texture Analysis for Image Segmentation.
  - IJCV 2001.
- **Mikolajczyk & Schmid**
  - Scale and Affine invariant interest point detectors.
  - IJCV 2004.
- **Moosmann *et al.***
  - Fast Discriminative Visual Codebooks using Randomized Clustering Forests.
  - NIPS 2006.
- **Nister & Stewenius**
  - Scalable Recognition with a Vocabulary Tree.
  - CVPR 2006.
- **Özuysal *et al.***
  - Fast Keypoint Recognition in Ten Lines of Code.
  - CVPR 2007.
- **Rogez *et al.***
  - Randomized Trees for Human Pose Detection.
  - CVPR 2008.
- **Sharp**
  - Implementing Decision Trees and Forests on a GPU.
  - ECCV 2008.
- **Shotton *et al.***
  - Semantic Texton Forests for Image Categorization and Segmentation.
  - CVPR 2008.
- **Shotton *et al.***
  - TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context.
  - IJCV 2007.
- **Sivic & Zisserman**
  - Video Google: A Text Retrieval Approach to Object Matching in Videos.
  - ICCV 2003.
- **Tibshirani & Hastie**
  - Margin trees for high-dimensional classification.
  - JMLR 2007.
- **Torralba *et al.***
  - Sharing visual features for multiclass and multiview object detection.
  - PAMI 2007.
- **Tu**
  - Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering.
  - ICCV 2005.
- **Tu**
  - Auto-context and Its application to High-level Vision Tasks.
  - CVPR 2008.
- **Tuytelaars & Schmid**
  - Vector Quantizing Feature Space with a Regular Lattice.
  - ICCV 2007.
- **Varma & Zisserman**
  - A statistical approach to texture classification from single images.
  - IJCV 2005.
- **Verbeek & Triggs**
  - Region Classification with Markov Field Aspect Models.
  - CVPR 2007.
- **Viola & Jones**
  - Robust Real-time Object Detection.
  - IJCV 2004.
- **Winn *et al.***
  - Object Categorization by Learned Universal Visual Dictionary.
  - ICCV 2005.
- **Wu *et al.***
  - Enlarging the Margins in Perceptron Decision Trees.
  - Machine Learning 2000.
- Yeh *et al.*
  - Adaptive Vocabulary Forests for Dynamic Indexing and Category Learning.
  - ICCV 2007.

Web Resources on Random Forests

- **Tutorial Webpage**
  - http://mi.eng.cam.ac.uk/~tkk22/iccv09_tutorial

- **Leo Breiman's Webpage**
  - http://www.stat.berkeley.edu/~breiman/RandomForests

- **Regression Trees**
  - http://www.stat.cmu.edu/~cshalizi/350-2006/lecture-10.pdf

End of Part I

Thank You

jamie@shotton.org

Internships at Microsoft Research Cambridge available for next spring/summer. Talk to me or see:

http://research.microsoft.com/en-us/jobs/intern/about_uk.aspx