Boosting & Randomized Forests for Visual Recognition

Jamie Shotton  Tae-Kyun Kim  Björn Stenger

Microsoft Research Cambridge  UNIVERSITY OF CAMBRIDGE  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

• 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]
[Shotton et al., 08]
[Criminisi et al., 09]
[Rogez et al., 08]
The Basics: *Is The Grass Wet?*

- **World state**
  - *Is it raining?*
    - no
      - *Is the sprinkler on?*
        - no
          - $P(\text{wet}) = 0.1$
        - yes
          - $P(\text{wet}) = 0.9$
    - yes
      - $P(\text{wet}) = 0.95$
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 \( P_n(c) \)

![Binary Decision Tree Diagram]

- \( f_1(\mathbf{v}) \leq t_1 \)
- \( f_3(\mathbf{v}) \leq t_3 \)
- \( f_6(\mathbf{v}) \leq t_6 \)
- \( f_{10}(\mathbf{v}) \leq t_{10} \)

- Leaf nodes
- Split nodes

category \( c \)
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)
        }
    } else {
        return node.P
    }
}
Toy Learning Example

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

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

• Recursively split examples at node $n$
  – set $I_n$ indexes labeled training examples $(v_i, l_i)$:

$$I_{l} = \{ i \in I_n \mid f(v_i) < t \}$$

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

• At node $n$, $P_n(c)$ is histogram of example labels $l_i$
More Randomized Learning

- Features $f(v)$ chosen at random from feature pool $f \in F$

- Thresholds $t$ chosen in range $t \in (\min_i f(v_i), \max_i f(v_i))$

- Choose $f$ and $t$ to maximize gain in information

$$\Delta E = -\frac{|I_1|}{|I_n|} E(I_1) - \frac{|I_r|}{|I_n|} E(I_r)$$

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
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

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
  \[2^d \times F \times T \times C\]
- Good for
  - very large data sets
  - shallower trees
GPU Acceleration

- 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 ]
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.
With a depth 2 tree, you cannot separate all six classes.
With a depth 3 tree, you are doing better, but still cannot separate all six classes.
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.
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

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

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

E.g. linear model $y = ax + b$, or just constant model
- Output is cluster membership

- Option 1 – minimize imbalance: \[ B = |\log|S_1|| - |\log|S_2|| \]  \[\text{[Moosmann et al. 06]}\]

- 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

- 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

---

[Moosmann et al. 06]

[Sivic et al. 03]

[Csurka et al. 04]
Clustering example

[Moosmann et al. 06]

tree $t_1$

......

tree $t_T$

“bag of words”
Relation to other parts of this tutorial

• **Boosting (Part II)**
  – decision trees as weak learners
  – boosted classifiers as split functions

  

• **Online learning (Part III)**
  – trees can be updated ‘online’
    • distributions of leaves
    • structure of tree

  [Yeh et al. 07]

[Tu 05]
Relation to Cascades

• **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
  – split functions have built-in generalization

• Tree of max-margin classifiers (SVMs)
  – recursively partition set of classes down the tree

[Wu et al., 00]

[Tibshirani & Hastie, 07]
Random Ferns

- Naïve Bayes classifier over random sets of features

\[
P(C|f_1, \ldots f_N) \propto P(f_1, \ldots f_N|C)P(C) \quad \text{Bayes’ rule}
\]

\[
\approx \prod_{j=1}^{N} P(f_j|C) \quad \text{“naïve Bayes”}
\]

individual features

\[
\approx \prod_{k=1}^{M} P(F_k|C) \quad \text{“random ferns”}
\]

set of features

- 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
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

classification algorithm
- SVM, decision forest, boosting
Object Recognition Pipeline

Semantic Texton Forest (STF)
- decision forest for clustering & classification
- tree nodes have learned object category associations

classification algorithm
SVM, decision forest, boosting
Example Semantic Texton Forest

Input Image

A[\text{g}] - B[\text{b}] > 28

[|A[\text{b}] - B[\text{g}]| > 37]

A[\text{b}] > 98

A[\text{r}] + B[\text{r}] > 363

A[\text{b}] + B[\text{b}] > 284

|A[\text{r}] - B[\text{b}]| > 21

A[\text{g}] - B[\text{b}] > 13

Example

Patches

P(c|l)

Ground Truth
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)
Segmentation Forest

• Object segmentation

• Adapt TextonBoost [Shotton et al. 07]
  – boosted classifier ➔ randomized decision forest
  textons ➔ semantic textons
MSRC Dataset Results
3D Point-Cloud Features

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

---

test image  | ground truth  | result
Human Pose Estimation

- 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

• Quickly localize bodily organs in 3D CT scans

Decision Forests with Long-Range Spatial Context for Organ Localization in CT Volumes

A. Criminisi, J. Shotton and S. Bucciarelli
Microsoft Research Ltd, Cambridge, UK

Brain Segmentation

[Yi et al. MICCAI 09]

ground truth

result

<table>
<thead>
<tr>
<th>Method</th>
<th>CSF</th>
<th>GM</th>
<th>WM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive MAP</td>
<td>0.069</td>
<td>0.564</td>
<td>0.567</td>
</tr>
<tr>
<td>Biased MAP</td>
<td>0.071</td>
<td>0.558</td>
<td>0.562</td>
</tr>
<tr>
<td>Fuzzy c-means</td>
<td>0.048</td>
<td>0.473</td>
<td>0.567</td>
</tr>
<tr>
<td>Maximum-a-posteriori (MAP)</td>
<td>0.071</td>
<td>0.550</td>
<td>0.554</td>
</tr>
<tr>
<td>Maximum-likelihood</td>
<td>0.062</td>
<td>0.535</td>
<td>0.551</td>
</tr>
<tr>
<td>Tree-Structure k-means</td>
<td>0.049</td>
<td>0.477</td>
<td>0.571</td>
</tr>
<tr>
<td>MPM-MAP [11]</td>
<td>0.227</td>
<td>0.662</td>
<td>0.683</td>
</tr>
<tr>
<td>MAP with histograms</td>
<td>0.549 ± 0.017</td>
<td>0.814 ± 0.004</td>
<td>0.710 ± 0.005</td>
</tr>
<tr>
<td>Decision Forest Classifier</td>
<td>0.614 ± 0.015</td>
<td>0.838 ± 0.006</td>
<td>0.731 ± 0.007</td>
</tr>
</tbody>
</table>
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.
- Bosch et al.
  - Image Classification using Random Forests and Ferns.
- Breiman
  - Random Forests.
- Breiman et al.
  - Classification and Regression Trees
- Brotz et al.
  - Segmentation and Recognition using Structure from Motion Point Clouds.
  - ECCV 2008.
- Csurka et al.
  - Visual Categorization with Bags of Keypoints.
- Fuchs & Buhmann
  - Inter-Active Learning of Randomized Tree Ensembles for Object Detection.
- Geurts et al.
  - Extremely Randomized Trees.
- Grauman & Darrell
  - The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features.
  - ICCV 2005.
- Hua et al.
  - Discriminant Embedding for Local Image Descriptors.
- Jurie & Triggs
  - Creating Efficient Codebooks for Visual Recognition.
  - ICCV 2005.
- Lazebrink 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.
- Malik et al.
  - Contour and Texture Analysis for Image Segmentation.
- Mikolajczyk & Schmid
  - Scale and Affine invariant interest point detectors.
- Moosmann et al.
  - Fast Discriminative Visual Codebooks using Randomized Clustering Forests.
  - NIPS 2006.
- Nicol & Stewenius
  - Scalable Recognition with a Vocabulary Tree.
  - CVPR 2006.
- Ózuysal et al.
- Röger 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.
- Singh & Zisserman
  - ICCV 2003.
- Tlzibhrani & Hastle
  - Margin trees for high-dimensional classification.
- Torralba et al.
  - Sharing visual features for multiclass and multiview object detection.
- 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.
- Varma & Zisserman
  - A statistical approach to texture classification from single images.
- Verbeek & Triggs
  - Region Classification with Markov Field Aspect Models.
- Viola & Jones
  - Robust Real-time Object Detection.
- Winn et al.
  - Object Categorization by Learned Universal Visual Dictionary.
  - ICCV 2005.
- Wu et al.
  - Enlarging the Margins in Perceptron Decision Trees.
- Yeh et al.
Web Resources on Random Forests

• **Tutorial Webpage**
  - [http://mi.eng.cam.ac.uk/~tkk22/iccv09_tutorial](http://mi.eng.cam.ac.uk/~tkk22/iccv09_tutorial)

• **Leo Breiman’s Webpage**
  - [http://www.stat.berkeley.edu/~breiman/RandomForests](http://www.stat.berkeley.edu/~breiman/RandomForests)

• **Regression Trees**
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