Three Algorithms in Large Scale

14210240041 Gu Pan

Overview

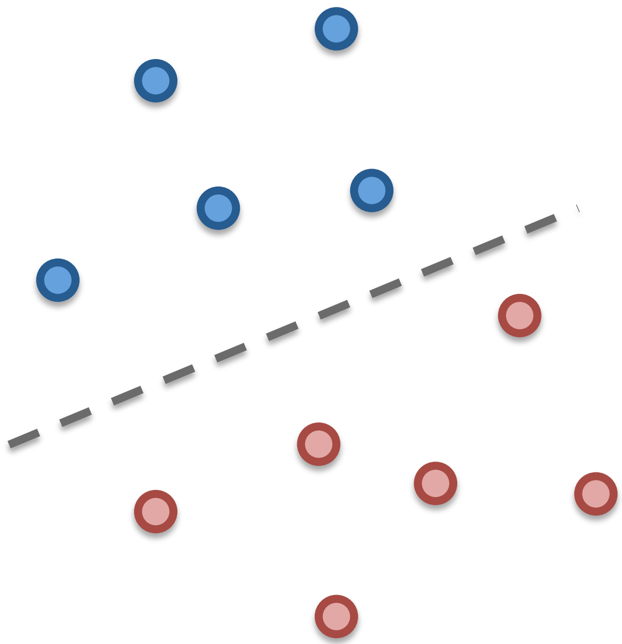
- More and more images have appeared on the Internet.

- Facebook, Twitter, Instagram…

Overview: Classification

- Classification is the problem of identifying to which of a set of categories a new observation belongs.

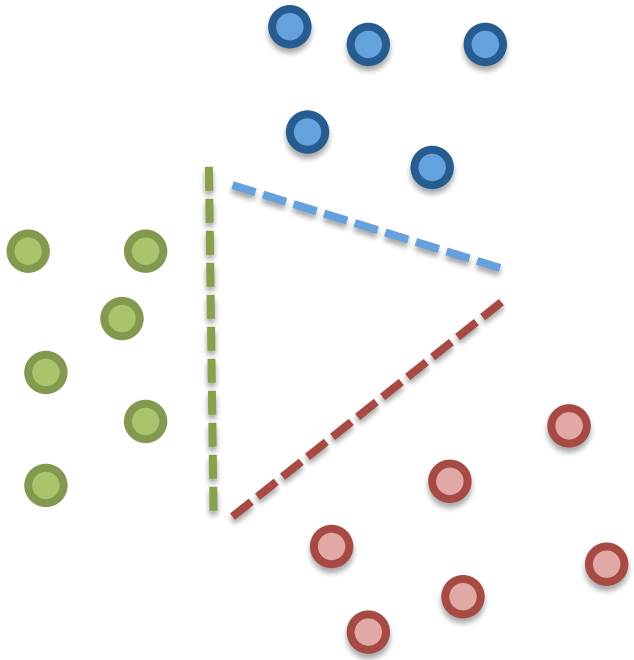
- Linear-SVM
- Kernel-SVM



Overview: SVM in Large Scale

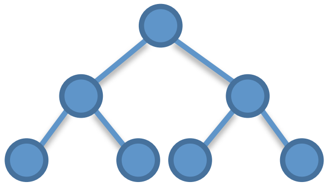
- Large image number
- Large category number

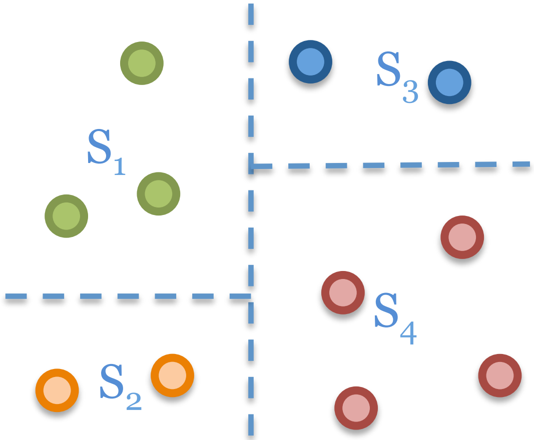
- Complexity: O(N*M)



Overview: Tree Structure Classifier

- Each non-leaf node of decision tree splits the feature space into two parts.
- Each leaf node of decision tree is labeled with a category.
- Complexity: $O(N*\log_2 M)$

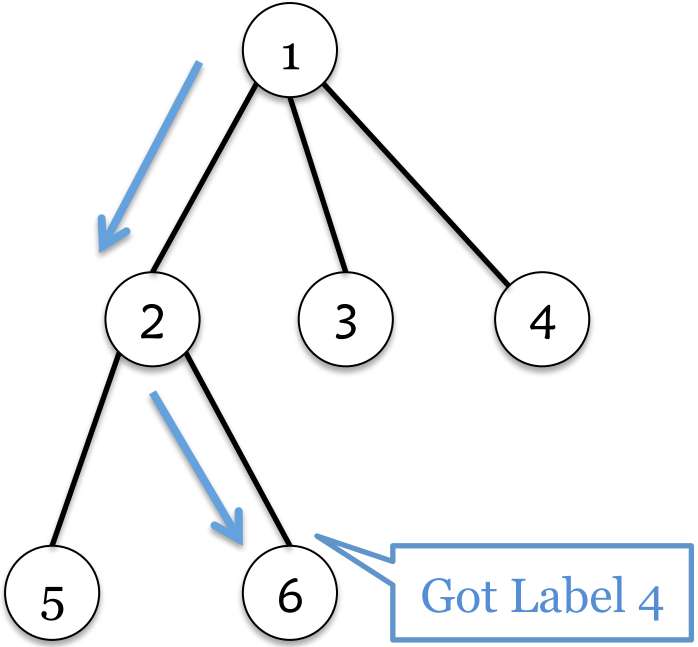




Label Tree

Label Embedding Trees for Large Multi-Class Tasks, NIPS 2010

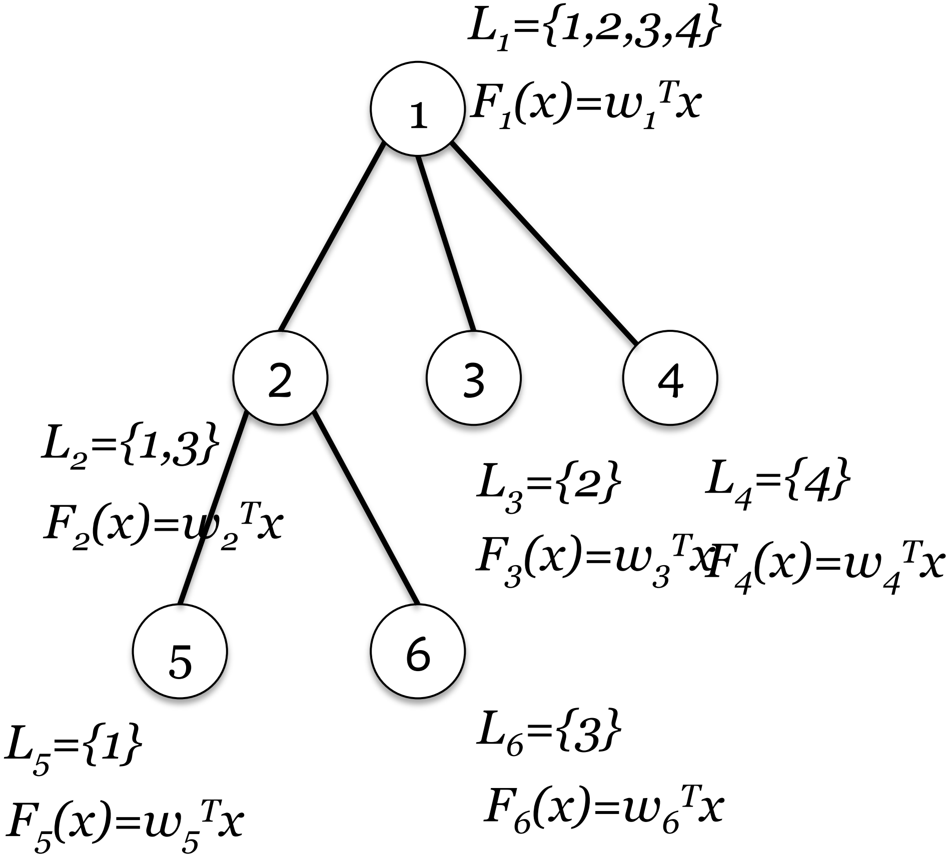
- Each node makes a prediction of the subset of labels to be children, thus decreasing the number of labels *k* at a logarithmic rate until a prediction is reached.



Label Tree: Algorithm Introduction

- Each node has a
  - label set L_i
  - predict function F_i

- Learning tree structure
- Learning parameter w_i for each node



Label Tree: Learning Tree Structures

- Learning structures for each node from its label set
  1. Calculate confusion matrix C from label set
  2. Using *spectral clustering* solving graph cut problem
  3. Create child node by label set split result
  4. Repeat 1-3 for each node

Label Tree: Confusion Matrix

- Method in paper:
  - $C_{ij} = \left|\{(x, y_i) \in V : argmax_r \bar{f}_r(x) = j\}\right|$ *on validation set V*
  - Non-robust
- Sigmoid method:
  - $C_{ij} = \sum_{(x,y_i)\in V} \frac{1}{1+e^{\bar{f}_j(x)}}$ *on validation set V*

Sample	$f_1(x)$	$f_2(x)$
$(x_1,1)$	0.51	0.49
$(x_2,1)$	0.51	0.49
$(x_3,2)$	0.49	0.51
$(x_4,2)$	0.49	0.51

Label Tree: Spectral Clustering

- Graph cut for confusion matrix C
  - $\min \ Cut(A,B) = \sum_{i\in A, j\in B} C_{ij}$
  - Unbalance for some case
- Normalized cut
  - $\min \ NCut(A,B) = \frac{Cut(A,B)}{\sum_{i\in A} C_{ij}} + \frac{Cut(A,B)}{\sum_{i\in B} C_{ij}}$

Label Tree: Relaxation 1

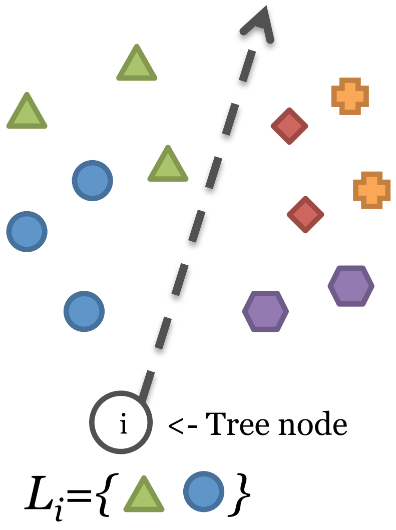
- Independent convex problems
- Learning parameter for each node indenpently

$$\min \sum_{j=1}^{n} \left(\gamma \left\| w_j \right\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_{ij} \right)$$

$$s.t. \forall i, j, C_j(y_i) f_j(x_i) \geq 1 - \xi_{ij}$$

- ξ_{ij} is slack variables

  $C_j(y)=1$ if $y \in L_i$ and -1 otherwise



Label Tree: Relaxation 2

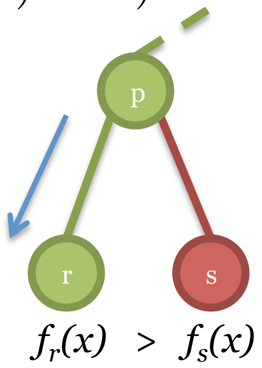
- Tree Loss Optimization
- Learning parameters for all node in a whole

$$\min \gamma \sum_{j=1}^{n} \|w_j\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i$$

$$s.t. f_r(x_i) \geq f_s(x_i) - \xi_i, \forall r, s : y_i \in l_r \wedge y_i \notin l_s \wedge (\exists p : (p, r) \in E \wedge (p, s) \in E)$$

$$\xi_i \geq 0, i = 1, \ldots, m$$

- ξ_i is slack variables.



Label Tree: Results of 1 & 2

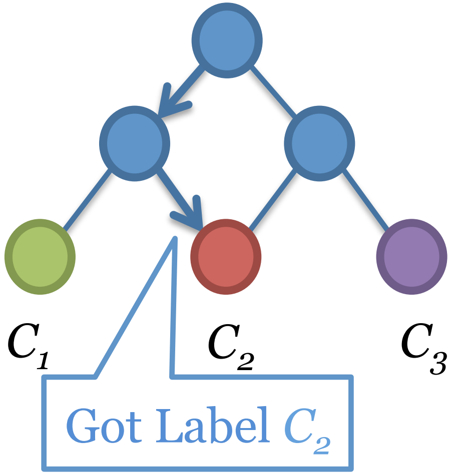
Method	Training Set Accuracy	Testing Set Accuracy
Relaxation 1	0.98 (12510/12800)	0.54 (4171/7680)
Relaxation 2	0.93 (11918/12750)	0.48 (3968/7650)

- Experiments on *Caltech256*
- Relaxation 2 has more parameters need to be adjusted
- Relaxation 1 has better effect
- Experiment code available on Github: https://github.com/gugugupan/LabelTree

Relaxed Hierarchy(RH)

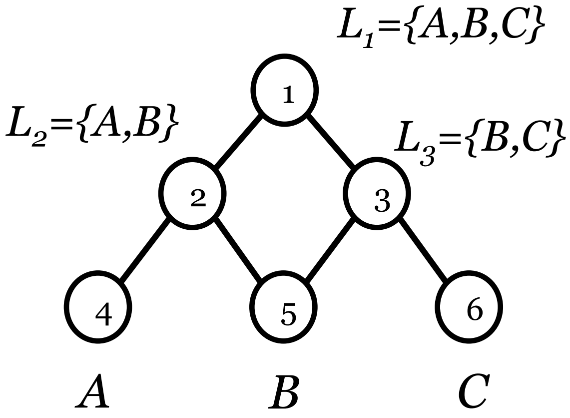
Constructing Category Hierarchies for Visual Recognition, ECCV2008

- Relaxed Hierarchy is based on the observation that finding a feature-space partitioning that reflects the class-set partitioning becomes more and more difficult with a growing number of classes



RH: Algorithm Introduction

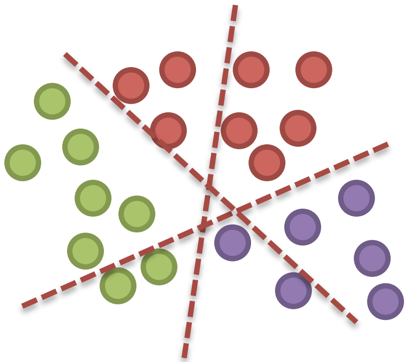
- Irregular non-leaf node counts
  - *DAG classifier*
  - Number of leaf node = Number of categories



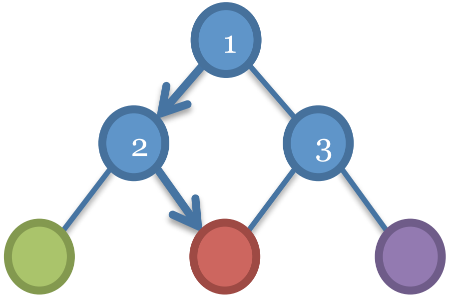
- Label set for each node
- Predict function for each non-leaf node

RH: Special Cases

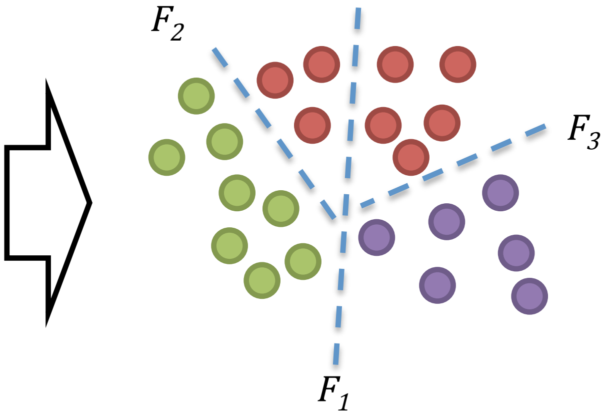
- Relaxed hierarchy can split non-linear case



Non-linear case

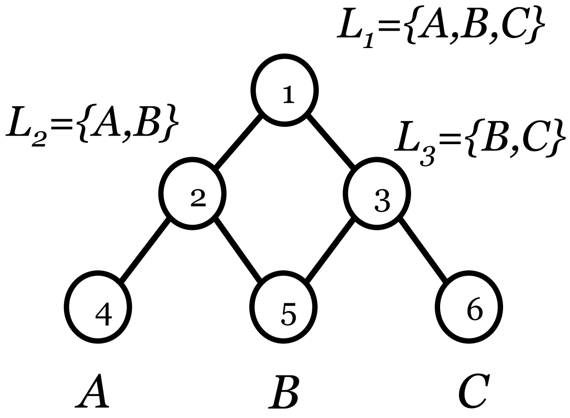


Relaxed Hierarchy



RH: Training(1)

- L_i is the label set for node i
  - Split L_i into 3 parts L, R and X
  - $L_i = L \lor R$
  - $X = L \land R$



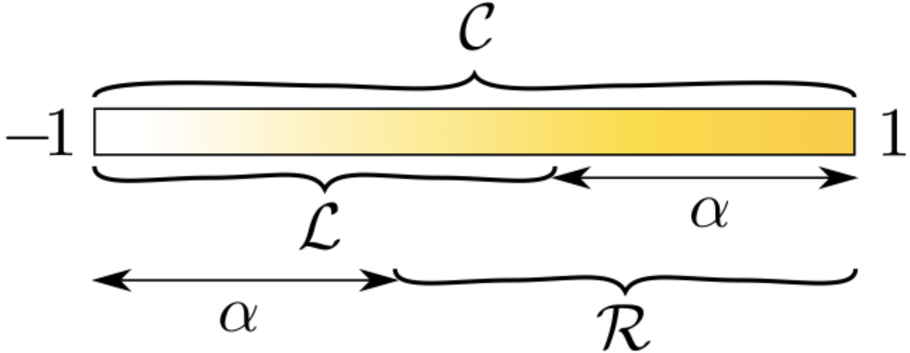
For node 1: L={A,B}, R={B,C}, X={B}

RH: Training(2)

- Define function: $$q(c) = \frac{1}{|\{(x,y)|y=c\}|} \sum_{(x,y),y=c} f(x)$$
  - c is a label in L_i
  - *f(x)* is the partition function by K-means
  - $f(x) = 1$ if $x \in Cluster_1$, and -1 otherwise
  - *q(c)* means the confidence category *c* belongs to *L* or *R*

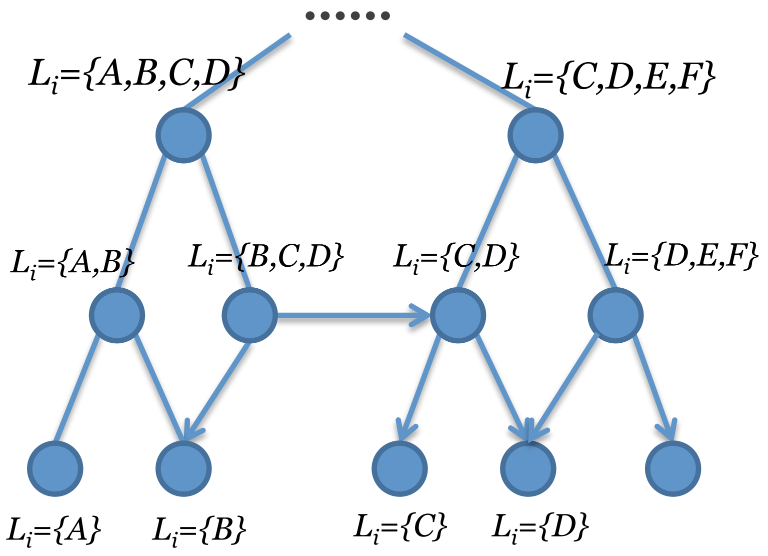
RH: Training(3)

- With function *q(c)*
  - $L = q^{-1}([-1,1-\alpha])$
  - $R = q^{-1}([-1+\alpha,1])$
  - where q^{-1} denotes an inverse image and α is a softening parameter

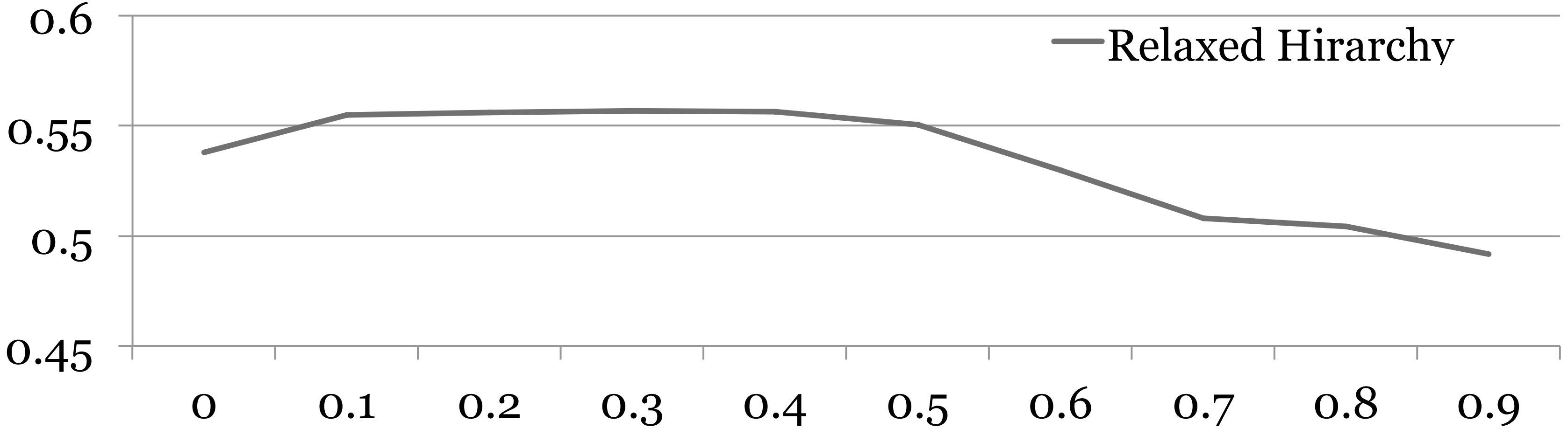


RH: Training(4)

- Label set of each node is its identification
  - Hashing or other index algorithm using here to find child node



RH: Results of different α



- Experiments on *Caltech256*
- Experiment code available on Github: https://github.com/gugugupan/RelaxedHierarchy

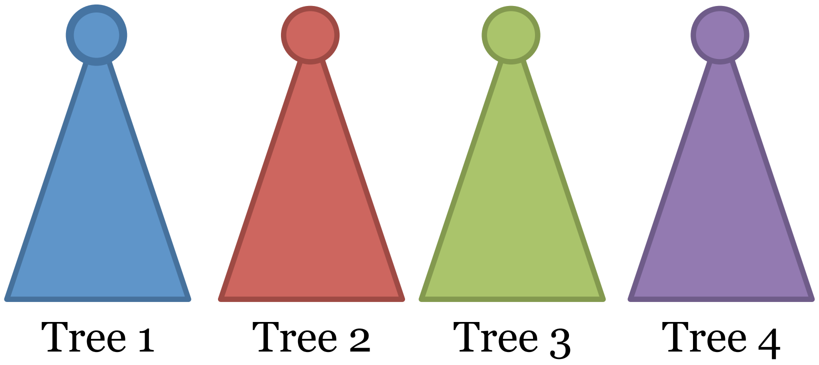
Random Forest(RF)

Random Forests, Machine Learning 45 (1): 5–32, 2001

- Random forests are an ensemble learning meth-od for classification (and regression) that opera-te by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes output by individual trees.

RF: Algorithm Introduction

- Bagging of random decision tree
- $\hat{f} = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(x')$

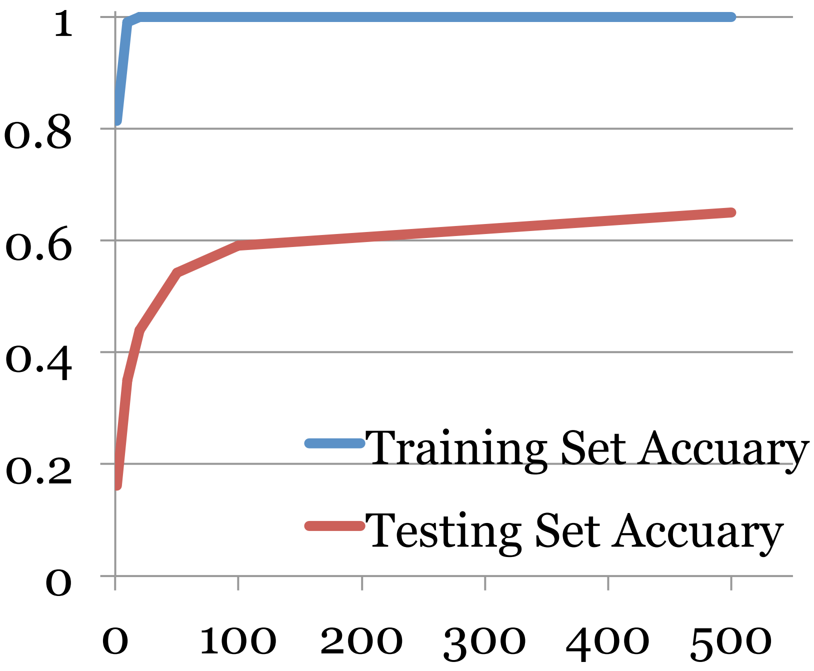


RF: Training

- Choose num of tree *B*, for each tree
  1. Select *N* bootstrap sample for training set
  2. For each node, random choose $m(\ll D)$ dimension subspace and find the best way to split sample in this subspace(as usual choose 1 dimension for split)
  3. Repeat 2 until leaf node

RF: Increase of Tree Num

- Experiments on *Caltech256*
- RF in training set has good fitting degree
- RF in testing set will conve-rge to some value
- Experiment code on: https://code.google.com/p/randomforest-matlab/ by abhirana



References

- Bengio S, Weston J, Grangier D. Label embedding trees for large multi-class tasks[C]//Advances in Neural Information Processing Systems. 2010: 163-171.
- Marszałek M, Schmid C. Constructing category hierarchies for visual recognition[M]//Computer Vision–ECCV 2008. Springer Berlin Heidelberg, 2008: 479-491.
- Breiman L. Random forests[J]. Machine learning, 2001, 45(1): 5-32.

Thanks