LOM  tree algorithm:

1.Introduction

**Problem:** the  number of classes k for multi-class prediction is  too large.

**Goal:** Try to find a algorithm with O(log(k)) time complexity for each example because of the Shannon entropy.

2.Framework and theoretical analysis

**data structure:** decision tree structure.

**the depth of tree:** log(k)

**classification:**

non-leaf: each node consists of a classifier from H.

leaf: each leaf, we predict according to the label with the highest frequency amongst the  examples reaching that leaf.

**performance:**

fairly balanced:  better computational complexity

fairly pure: better statistical accuracy.

traditional way to use the Shannon or Gini entropy to check if the partition at a fixed node is good or not.

for online problem, we need to define a new objective function to measure  the  quality of a hypothetic classifier from H in creating partitions at a fixed node n in the tree.

**Goal:** maximize J(h) to get better partition(balanced and pure)

partition criterion: E(h(x))-E(h(x)|y)>0, then the data  point goes to the left child node, else to the right child node.

**algorithm:**

while generating a tree, we meet two cases:

1.current node j is a leaf node:

    a. the current node now has at least two non-zero entries(over 2-classes labels)

    b. the number of non-leaf nodes< threshold T

             new left child node and right child node

2.current  node j is not  a leaf node:

    use splitting criterion(E(h(x))-E(h(x)|y)>0), if(>0) c=-1, else c=1. Given each node unique label, then training the node classifier with example(x,c ) (notice: now the label becomes c not y), update ej and Ej for  node j, then set the  pointer j to the  child of j corresponding to hj.

For each node:

create a class(node):

class\_name: [] (list to store each label of data going through the node)

left, right children, parent

node:  classifier(mode) : logistice regression + sigmoid func

expectation: Ej and ej (use Ej and ej to check the node should go to left child or right child, then update the expectation)

tree use list[] to store all the nodes, use

swap: