Final Project Proposal
Fast Ada-boost Using Heuristic Decision Trees

Ze Wang

March-31-2013

Abstract

AdaBoost, short for Adaptive Boosting, is a meta machine learning algorithm that can be used in conjunction with many other learning algorithms to improve their performance.Its main idea is to use weak learner to create strong learners and subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifier.One kind of weak classifiers is decision tree that uses some heuristics to find suboptimal trees such as using information gain or gini impurity.This project aims at implementing a fast ada-boost library based on decision tree classifiers.

1 Introduction

A concept class C is weakly PAC-learnable if there exists a (weak) learning algorithm L and $\gamma > 0$ such that: for all $c \in C, \epsilon > 0, \sigma > 0$ and all distributions d:

$$\Pr_{S \sim D}[R(hs) \leq \frac{1}{2} - \gamma] \geq 1 - \sigma$$

for samples S of size $m = poly(\frac{1}{\sigma})$ for a fixed polynomial. Adaboost is based on this guarantee to combine a bunch of weak classifiers. This project will implement logistic Adaboost over a top-down decision tree according to [3].I will use C++ to write the whole library and [1] as the experiment data.

2 Heuristic Decision Tree

The most popular decision tree implementation is C4.5 and ID3 algorithms. Unfortunately they are not suitable for a fast decision tree growing. Standard implementation of C4.5 algorithm will have a time complexity of $O(m \cdot n^2)$ where m is the number of training examples and n is the number of attributes.By applying some conditional independence assumption inspired by Naive Bayes, we can develop a fast decision tree algorithm [3] that can run at $O(m \cdot n)$ without losing too much accuracy.

Most modern decision tree learning algorithm will adopt a purity-based heuristic that measures the purity of the resulting subsets after partitioning the data by the splitting attributes.E.g. ID3 , C4,5 use the information gain to find the best attribute, where the information gain is defined as:

$$I_G(S, X) = Entropy(S) - \sum_x \frac{|S_x|}{S} Entropy(S_x)$$

where S is the set of training examples, X is any attribute and x is its value, and Entropy(S) is defined as

$$Entropy(S) = -|\sum_{i=1}^{|C|} P_S(c_i) log P_s(c_i),$$

where $|C|$ is the number of classes and $P_S(c_i)$ is the ratio of number of examples belonging to c_i in S.

Entropy measures the uncertainty in a random variables.So generally the attribute that leads to

a biggest information gain will cause the most significant decrease the entropy of current subset of examples. This tree growing is a recursive process of partitioning the training data and S is associated with the current node. The $P_S(c_i)$ is actually $P(c_i|x_p)$ where x_p is the attributes along the path from root to current node and similarly $P_{S_x}(c_i)$ is $P(c_i|x_p, x)$ on the entire training data.

So from here we can see the most time-consuming part is computing $P_{S_x}(c_i)$ since it must iterate through each candidate attribute X.
By observing

$$P(c_i|x_p, x) = \frac{P(c_i|x_p)P(x|x_p, c_i)}{P(x|x_p)} \quad (1)$$

$$= \frac{P(c_i|x_p)P(x|x_p, c_i)}{\sum_{i=1}^{|C|} P(c_i|x_p)P(x|x_p, c_i)} \quad (2)$$

Assume $P(x|x_p, C) = P(x|C)$, this means each attribute is independent of the path attribute assignment x_p give the class.
Then we have

$$P(c_i|x_p, x) \approx \frac{P(c_i|x_p)P(x|c_i)}{\sum_{i=1}^{|C|} P(c_i|x_p)P(x|c_i)}$$

The information gain computed by this equation is called *Independent Information Gain*(IIG).
We noticed that $P(x|c_i)$ can be precomputed and store with a time complexity of $O(m \cdot n)$ and $P(c_i|x_p)$ can be computed by passing through S which takes $O(|S|)$ so at each level, the time complexity for computing $P(c_i|x_p, x)$ is $O(m)$, thus the overall complexity is $O(m \cdot n)$

3 Adaboost Implementation

In this project I will implement the AdaBoost according to [2].It's a generalized implementation of Adaboost where we just update

$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution, $\alpha_t = \frac{1}{2}ln(\frac{1+r_t}{1-r_t}), Z_t = 2[r_t(1-rt)]^{\frac{1}{2}}$ The final result is $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

4 Plan

1. Survey:April 5 - April 10
   I will look into some related works about how Adaboost is implemented, what kind of heuristic that decision tree can use that can make it run fast without losing too much accuracy.
2. Implementation:April 11 - April 24

   In this stage I will implement the whole algorithm that:

   - Ultimate Goal: Implement a fast adaboost library taking advantage of the weak guarantee and using fast heuristic decision tree as the underlying weak classifier.
   - Input: A set of training instances of which the attributes values are real numbers for training and a set of instance with unknown labels.
   - Output: Predicated label of each of the unknown instances
   - Implemented language: C++ with OpenMP
3. Experiment:April 24 - May 1
   I will conduct the experiments using the data [1].
4. Presentation and Report:May 21

5 Experiment

I will use [1] as my experiment data.It's Element-wise comparison of records with personal data from a record linkage setting whose

goal is to decide from a comparison pattern whether the underlying records belong to one person. It has 5749132 instances and 12 attributes.Each of which is a real number and it contains no missing value. However originally, this dataset is for multivariate classification thus we need modify the adaboost a little to adapt into this situation. Also I will compare the accuracy and run time of my algorithm with those of the Vowpal Wabbit.

References

[1] Record linkage comparison patterns data set.

[2] Schapire and Singer. Improved boosting algorithms usingon fidence-rated predictions. *Machine Learning*, 37(E2):297–336, 1999.

[3] Jiang Su and Harry Zhang. A fast decision tree learning algorithm.