COT6930: Data Mining
Meta Learning Schemes
Bagging, Boosting, CostBoosting

Erik Geleyn

Empirical Software Engineering Laboratory
Dept. of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
(561)297-2512
egeleyn@cse.fau.edu
http://www.cse.fau.edu/esel.html

Overview

Introduction to Meta Learning Schemes

Bagging

Boosting

CostBoosting

Summary

Introduction to Meta Learning Schemes

- Foreword
- Concepts
- Advantages
- Drawbacks

Introduction to Meta Learning Schemes

Foreword

- Combining several learners
- Analogy with some human decision process

Introduction to Meta Learning Schemes

Concepts

- What kind of classifier should be combined?
- Stable and unstable learners
- Stable: CBR, linear regression
- Unstable: Decision Trees, neural nets
- Weak and Strong learners (Decision Stump, C4.5)

Introduction to Meta Learning Schemes

Advantages

- Performance
- Less overfitting
- No tuning

Introduction to Meta Learning Schemes

Drawbacks

- Combined decisions are harder to interpret
- Computational greediness

Bagging

- Concepts
- Algorithm
- Example

Bagging

Concepts

- Simple

- Builds different models by randomly resampling from the original training dataset

- Easy to implement on parallel architectures

- Able to improve weak learners

Bagging

Algorithm

Notations

Symbol	*Description*
h_t	Weak hypothesis on the t^{th} iteration
$h_t(x_i)$	Value of the weak hypothesis on instance i
$h_{fin}(x_i)$	Final hypothesis
m	Number of instances in training dataset
S	A Training dataset
T	Number of iterations
X	An instance space
x_i	An instance in an instance space X
Y	Class space
y_i	A class in Y

Bagging

Algorithm

Notations

1. **Input**:
   - A data set S of order pairs $(x_1, y_1), ..., (x_m, y_m)$, where $x_i \in X$ is an instance space, $y_i \in Y = \{-1, +1\}$
   - Weak learning algorithm
   - An integer T specifying the number of iterations
2. **Do for** $t = 1, 2, ..., T$
   - Form a data set S_t by sampling n instances with replacement from the training data set S
   - Call Weak Learner, providing it with the distribution S_t
   - Get back a hypothesis $h_t : X \rightarrow Y$.
3. **Output** the final Hypothesis: $h_{fin}(x_i) = sign(\sum_{t=1}^{T} h_t(x_i))$

Bagging

Example

Simple voting

x_i	y_i	$h_1(i)$	$h_2(i)$	$h_3(i)$	$h_4(i)$	$h_5(i)$	$h_{fin}(x_i)$
x_1	1	1	1	1	-1	1	
x_2	1	1	-1	1	1	1	
x_3	-1	-1	1	1	1	-1	
x_4	1	-1	1	1	1	1	
x_5	1	1	1	1	-1	1	
x_6	-1	1	-1	-1	-1	-1	
x_7	-1	-1	-1	-1	1	-1	
x_8	-1	-1	-1	1	-1	-1	
x_9	-1	-1	-1	-1	-1	-1	
x_{10}	1	1	1	1	1	1	

Boosting

- Concepts
- Algorithm
- Weighted Datasets
- Stochastic Sampling with Replacement
- Example

Concepts

- Boosting VS. Bagging
- Uses previous misclassification history
- Uses a weighted dataset to generate the different models
- Increases performances in a more significant way than Bagging
- Still, sometimes can worsen a strong learner

Boosting

Algorithm

Notation

Symbol	*Description*
α_t	Parameter choosen as a weight for weak hypothesis h_t
$D_t(i)$	Distribution used as a weight for instance i at iteration t
$D_{t+1}(i)$	Distribution used as a weight for instance i at iteration $t+1$
ϵ_t	Error of the weak hypothesis h_t
h_t	Weak hypothesis on the t^{th} iteration
$h_t(x_i)$	Value of the weak hypothesis on instance x_i
$h_{fin}(x_i)$	Final hypothesis
m	Number of instances in training data set
T	Number of iterations
X	An instance space
x_i	An instance in an instance space X
Y	Class space
y_i	A class in Y
Z_t	Normalization constant to ensure that D_{t+1} will be a distribution

Boosting

Algorithm

---

1. **Input**:
   - A set of order pairs $(x_1, y_1), ..., (x_m, y_m)$, where $x_i \in X$ is an instance space, $y_i \in Y = \{-1, +1\}$
   - Weak learning algorithm
   - An integer T specifying the number of iterations
2. **Initialize** $D_1(i) = 1/m$ for all i.
3. **Do for** $t = 1, 2, ..., T$
   - Call Weak Learner, providing it with the distribution D_t
   - Get back a hypothesis $h_t : X \rightarrow Y$.
   - Calculate the error of h_t : $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$. If $\epsilon_t > \frac{1}{2}$, then set $T = t - 1$ and abort loop.
   - Set $\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t}$
   - Update distribution D_t : $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$ where Z_t is a normalization constant(chosen so that D_{t+1} will be a distribution).
4. **Output** the final Hypothesis: $h_{fin}(x_i) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x_i)\right)$

Boosting

Weighted Datasets

Two ways to use weights in a meta learning scheme:

1. If the algorithm allows it, the weights are used to build the preferred learner. Typically, the weights are used to compute the error of a learner.

2. Otherwise, we induce the weights by resampling form the original training dataset. Instances with higher weights are given a higher probability of being resampled.

Boosting

Stochastic Sampling with Replacement

- Concept: a spinning roulette with slots of different sizes

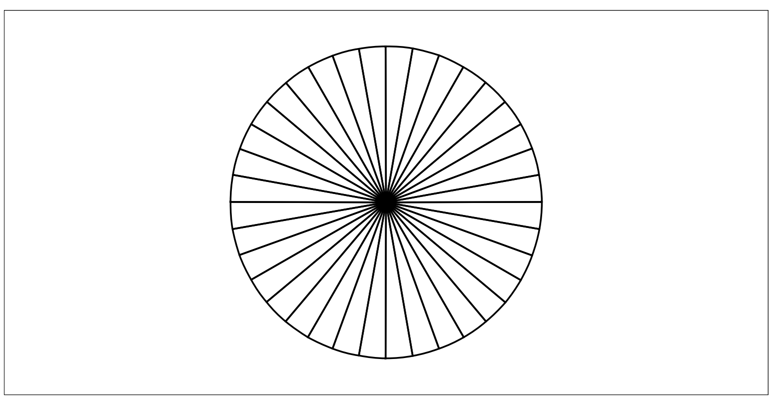
- Example: weight table

Instance	Weight	Slot Angle (degrees)
1	0.0555	
2	0.0278	
3	0.1111	
4	0.1111	
5	0.4444	
6	0.0278	
7	0.1111	
8	0.0555	
9	0.0278	
10	0.0278	

Boosting

Stochastic Sampling with Replacement

Spinning roulette



Random numbers: 65, 327, 48, 348, 128, 142, 230, 337, 11, and 106.

Resampled instances:

Boosting

Example

Weight updates:

$D_1(i)$	x_i	y_i	$h_1(i)$	$e^{\pm\alpha_1}$	$D_1(i) \times e^{\pm\alpha_1}$	$D_2(i)$
0.1000	x_1	1	1			
0.1000	x_2	-1	1			
0.1000	x_3	1	-1			
0.1000	x_4	1	-1			
0.1000	x_5	1	1			
0.1000	x_6	-1	1			
0.1000	x_7	-1	-1			
0.1000	x_8	-1	-1			
0.1000	x_9	-1	-1			
0.1000	x_{10}	1	1			
					z=	

CostBoosting

- Concepts
- Algorithm
- Example

Concepts

- Cost-Boosting VS. Boosting
- Specificity of Software Quality Modeling
- Inducing cost-sensitivity in the meta learning algorithm

CostBoosting

Algorithm

Notation:

Symbol	Description
α_t	Parameter chosen as a weight for weak hypothesis h_t
$cost(k, j)$	Misclassification cost of classifying a class k instance as class j
$D_t(i)$	Distribution used as a weight for instance i on iteration t
$D_{t+1}(i)$	Distribution used as a weight for instance i on iteration $t+1$
$D^{'}_{t+1}(i)$	Cost adjustment factor used to determine $D_{t+1}(i)$
h_t	Weak hypothesis on the t^{th} iteration
$h_t(x_i)$	Value of the weak hypothesis on instance x_i
$h_{fin}(x_i)$	Final hypothesis
K	Total number of classes
m	Number of instances in training data set
T	Number of iterations
X	An instance space
x_i	An instance in an instance space X
Y	Class space
y_i	A class in Y

CostBoosting

Algorithm

1. **Input**:
   - A set of order pairs $(x_1, y_1), ..., (x_m, y_m)$, where $x_i \in X$ is an instance space, $y_i \in Y = \{-1, +1\}$
   - Weak learning algorithm
   - An integer T specifying the number of iterations
2. **Initialize** $D_1(i) = 1/m$ for all i.
3. **Do for** $t = 1, 2, ..., T$
   - Call Weak Learner, providing it with the distribution D_t
   - Get back a hypothesis $h_t : X \rightarrow Y$.
   - Calculate the error of h_t : $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$. If $\epsilon_t > \frac{1}{2}$, then set $T = t - 1$ and abort loop.
   - Set $\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t}$
   - Update distribution D_t : $D_{t+1}(i) = \frac{D'_{t+1}(i)}{\sum_i^m D'_{t+1}(i)}$

     $D'_{t+1}(i) = \begin{cases} cost(actual(i), predicted(i)) & \text{if } actual(i) \neq predicted(i) \\ mD_t(i) & \text{otherwise.} \end{cases}$
4. **Output** the final Hypothesis: $h_{fin}(x_i) = min \sum_k^K \sum_{t=1}^T |\alpha_t h_t(x_i) cost(k, j)|$,

   where K is the total number of classes; and $cost(k, j)$ is the misclassification cost of classifying a class k instance as class j.

CostBoosting

Example

Weight updates:

i	y_i	$D_1(i)$	$h_1(i)$	$D'_2(i)$	$D_2(i)$	$h_2(i)$	$D'_3(i)$	$D_3(i)$
1	1	0.067	1			1		
2	1	0.067	-1			1		
3	1	0.067	1			1		
4	1	0.067	-1			-1		
5	1	0.067	-1			1		
6	1	0.067	1			-1		
7	1	0.067	1			1		
8	1	0.067	-1			-1		
9	1	0.067	1			1		
10	1	0.067	1			1		
11	-1	0.067	-1			-1		
12	-1	0.067	-1			-1		
13	-1	0.067	-1			1		
14	-1	0.067	1			-1		
15	-1	0.067	-1			-1		

Summary

- Increased performance
- Less prone to overfitting
- No tuning
- Ability to use previous misclassification history
- Cost-sensitive