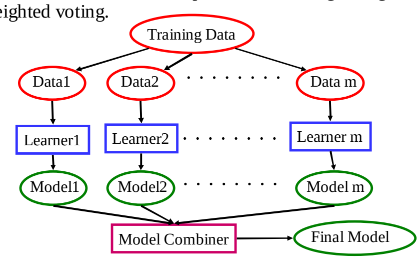
CS 391L: Machine Learning: Ensembles

Raymond J. Mooney
University of Texas at Austin

1

Learning Ensembles

- Learn multiple alternative definitions of a concept using different training data or different learning algorithms.
- Combine decisions of multiple definitions, e.g. using weighted voting.



2

Value of Ensembles

- When combing multiple ***independent*** and ***diverse*** decisions each of which is at least more accurate than random guessing, random errors cancel each other out, correct decisions are reinforced.
- Human ensembles are demonstrably better
  - How many jelly beans in the jar?: Individual estimates vs. group average.
  - Who Wants to be a Millionaire: Expert friend vs. audience vote.

3

Homogenous Ensembles

- Use a single, arbitrary learning algorithm but manipulate training data to make it learn multiple models.
  - Data1 ≠ Data2 ≠ … ≠ Data m
  - Learner1 = Learner2 = … = Learner m
- Different methods for changing training data:
  - Bagging: Resample training data
  - Boosting: Reweight training data
  - DECORATE: Add additional artificial training data
- In WEKA, these are called ***meta-learners***, they take a learning algorithm as an argument (***base learner***) and create a new learning algorithm.

4

Bagging

- Create ensembles by repeatedly randomly resampling the training data (Brieman, 1996).
- Given a training set of size *n*, create *m* samples of size *n* by drawing *n* examples from the original data, ***with replacement***.
  - Each ***bootstrap sample*** will on average contain 63.2% of the unique training examples, the rest are replicates.
- Combine the *m* resulting models using simple majority vote.
- Decreases error by decreasing the variance in the results due to ***unstable learners***, algorithms (like decision trees) whose output can change dramatically when the training data is slightly changed.

5

Boosting

- Originally developed by computational learning theorists to guarantee performance improvements on fitting training data for a ***weak learner*** that only needs to generate a hypothesis with a training accuracy greater than 0.5 (Schapire, 1990).
- Revised to be a practical algorithm, AdaBoost, for building ensembles that empirically improves generalization performance (Freund & Shapire, 1996).
- Examples are given weights. At each iteration, a new hypothesis is learned and the examples are reweighted to focus the system on examples that the most recently learned classifier got wrong.

6

Boosting: Basic Algorithm

- General Loop:

  Set all examples to have equal uniform weights.
  For t from 1 to T do:
  Learn a hypothesis, h_t, from the weighted examples
  Decrease the weights of examples h_t classifies correctly
- Base (weak) learner must focus on correctly classifying the most highly weighted examples while strongly avoiding over-fitting.
- During testing, each of the T hypotheses get a weighted vote proportional to their accuracy on the training data.

7

AdaBoost Pseudocode

TrainAdaBoost(D, BaseLearn)
For each example d_i in D let its weight w_i=1/|D|
Let H be an empty set of hypotheses
For t from 1 to T do:
Learn a hypothesis, h_t, from the weighted examples: h_t=BaseLearn(D)
Add h_t to H
Calculate the error, ε_t, of the hypothesis h_t as the total sum weight of the examples that it classifies incorrectly.
If $\varepsilon_t > 0.5$ then exit loop, else continue.
Let $\beta_t = \varepsilon_t \ / \ (1 - \varepsilon_t)$
Multiply the weights of the examples that h_t classifies correctly by β_t
Rescale the weights of all of the examples so the total sum weight remains 1.
Return H

TestAdaBoost(ex, H)
Let each hypothesis, h_t, in H vote for ex's classification with weight log(1/ β_t)
Return the class with the highest weighted vote total.

8

Learning with Weighted Examples

- Generic approach is to replicate examples in the training set proportional to their weights (e.g. 10 replicates of an example with a weight of 0.01 and 100 for one with weight 0.1).
- Most algorithms can be enhanced to efficiently incorporate weights directly in the learning algorithm so that the effect is the same (e.g. implement the WeightedInstancesHandler interface in WEKA).
- For decision trees, for calculating information gain, when counting example i, simply increment the corresponding count by w_i rather than by 1.

9

Experimental Results on Ensembles
(Freund & Schapire, 1996; Quinlan, 1996)

- Ensembles have been used to improve generalization accuracy on a wide variety of problems.
- On average, Boosting provides a larger increase in accuracy than Bagging.
- Boosting on rare occasions can degrade accuracy.
- Bagging more consistently provides a modest improvement.
- Boosting is particularly subject to over-fitting when there is significant noise in the training data.

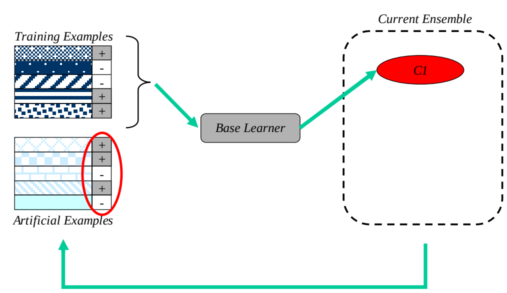
10

DECORATE
(Melville & Mooney, 2003)

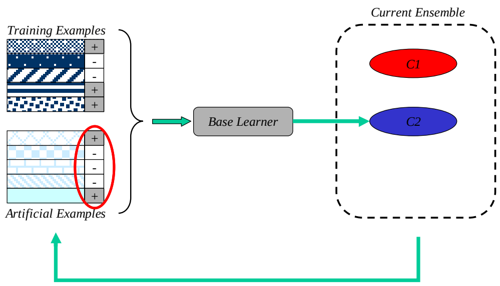
- Change training data by adding new artificial training examples that encourage diversity in the resulting ensemble.
- Improves accuracy when the training set is small, and therefore resampling and reweighting the training set has limited ability to generate diverse alternative hypotheses.

11

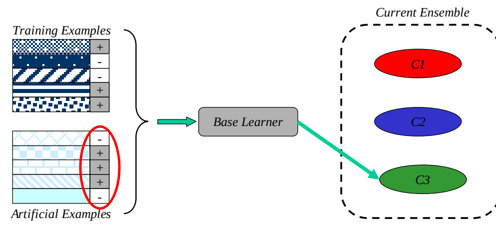
Overview of DECORATE



Overview of DECORATE



Overview of DECORATE

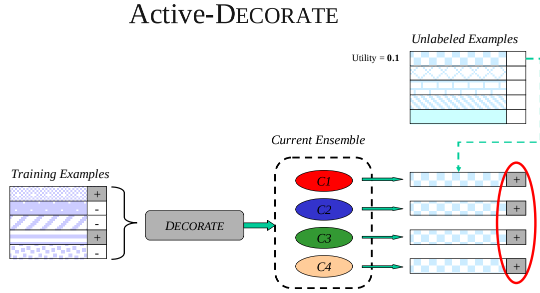


Ensembles and Active Learning

- Ensembles can be used to actively select good new training examples.
- Select the unlabeled example that causes the most disagreement amongst the members of the ensemble.
- Applicable to any ensemble method:
  - QueryByBagging
  - QueryByBoosting
  - ActiveDECORATE

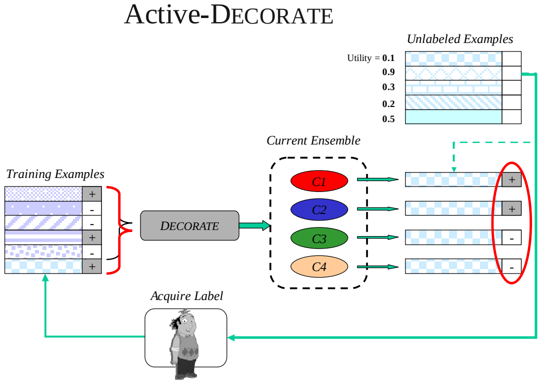
15

Active-DECORATE



16

Active-DECORATE



17

Issues in Ensembles

- Parallelism in Ensembles: Bagging is easily parallelized, Boosting is not.
- Variants of Boosting to handle noisy data.
- How “weak” should a base-learner for Boosting be?
- What is the theoretical explanation of boosting’s ability to improve generalization?
- Exactly how does the diversity of ensembles affect their generalization performance.
- Combining Boosting and Bagging.

18