Ensemble Learning

CS 6243 Machine Learning

Modified from the slides by Dr. Raymond J. Mooney
http://www.cs.utexas.edu/~mooney/cs391L
And by Dr. Carla P. Gomes
http://www.cs.cornell.edu/Courses/cs4700/2008fa/

Ensemble Learning

So far – learning methods that learn a single hypothesis, chosen form a hypothesis space that is used to make predictions.

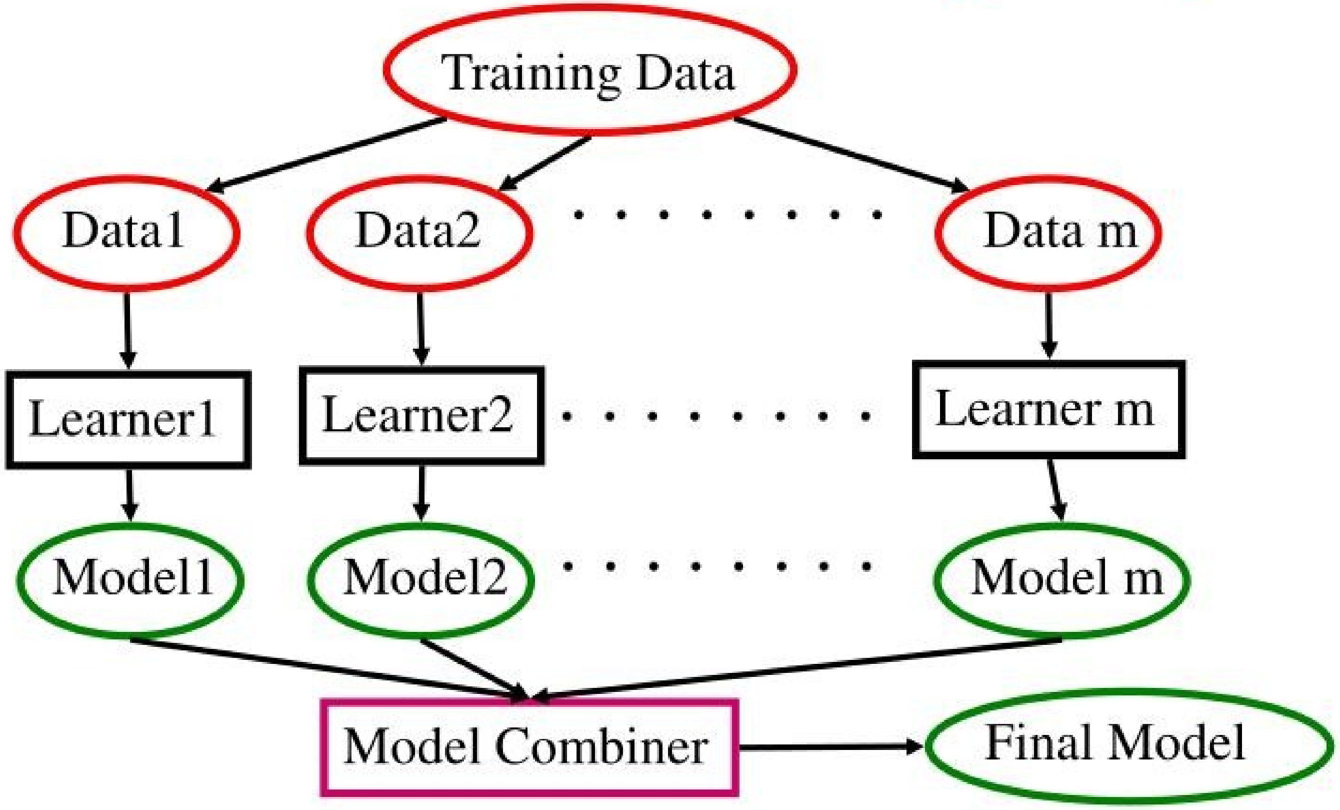
Ensemble learning → select a collection (ensemble) of hypotheses and combine their predictions.

Example 1 - generate 100 different decision trees from the same or different training set and have them vote on the best classification for a new example.

Key motivation: reduce the error rate. Hope is that it will become much more unlikely that the ensemble of will misclassify an example.

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.



Source: Ray Mooney

Value of Ensembles

“No Free Lunch” Theorem

– No single algorithm wins all the time!

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.

Source: Ray Mooney

Example: Weather Forecast

Reality	⛈	☀	☀	⛈	⛈	☀	☀
1	⛈	⛈	☀	☀	⛈	☀	⛈
2	☀ ✗	☀ ✗	☀	☀ ✗	⛈	☀	⛈ ✗
3	⛈	☀	⛈ ✗	⛈	☀ ✗	⛈ ✗	☀
4	⛈	☀	⛈ ✗	⛈	☀ ✗	☀	☀
5	⛈	⛈ ✗	☀	⛈	⛈	⛈ ✗	☀
Combine	⛈	☀	☀	⛈	⛈	☀	☀

Intuitions

Majority vote

Suppose we have 5 completely independent classifiers…

- If accuracy is 70% for each
  - $(.7^5)+5(.7^4)(.3)+10(.7^3)(.3^2)$
  - **83.7% majority vote accuracy**
- 101 such classifiers
  - **99.9% majority vote accuracy**

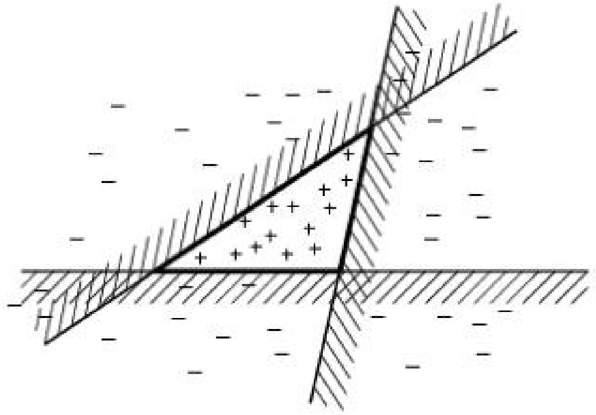
Note: Binomial Distribution: The probability of observing *x* heads in a sample of *n* independent coin tosses, where in each toss the probability of heads is *p*, is

$$P(X=x|p,n)=\frac{n!}{r!(n-x)!}p^x(1-p)^{n-x}$$

Ensemble Learning

Another way of thinking about ensemble learning:

→ way of enlarging the hypothesis space, i.e., the ensemble itself is a hypothesis and the new hypothesis space is the set of all possible ensembles constructible form hypotheses of the original space.



Increasing power of ensemble learning:

Three linear threshold hypothesis
(positive examples on the non-shaded side);
Ensemble classifies as positive any example classified positively be all three. The resulting triangular region hypothesis is not expressible in the original hypothesis space.

Different types of ensemble learning

Different learning algorithms

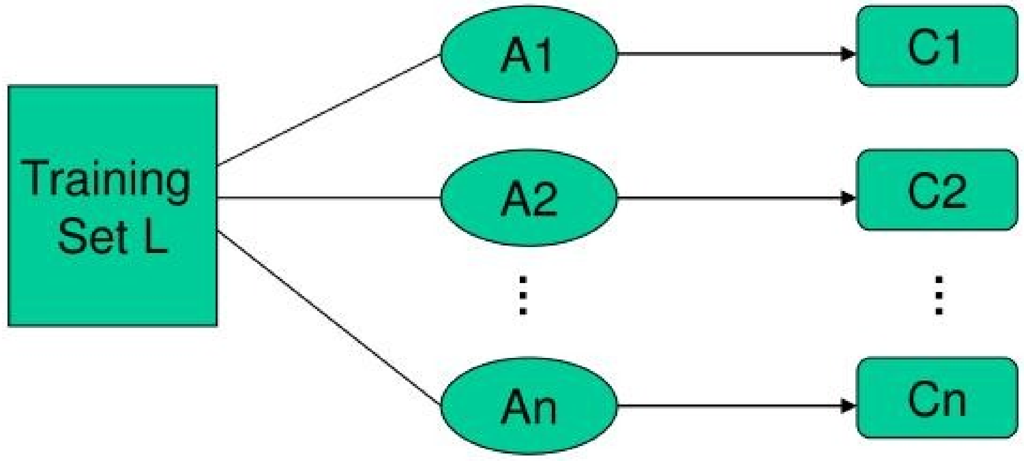
Algorithms with different choice for parameters

Data set with different features (e.g. random subspace)

Data set = different subsets (e.g. bagging, boosting)

Different types of ensemble learning (1)

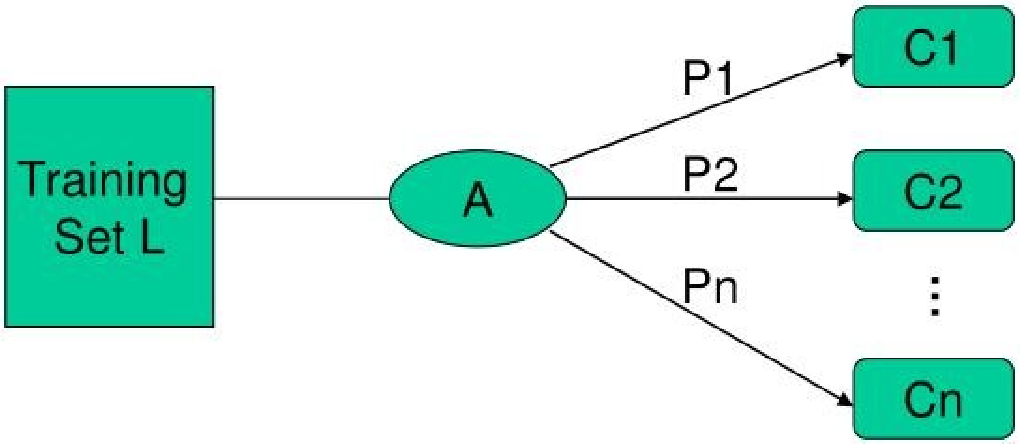
Different algorithms, same set of training data



A: algorithm I; C: classifier

Different types of ensemble learning (2)

Same algorithm, different parameter settings

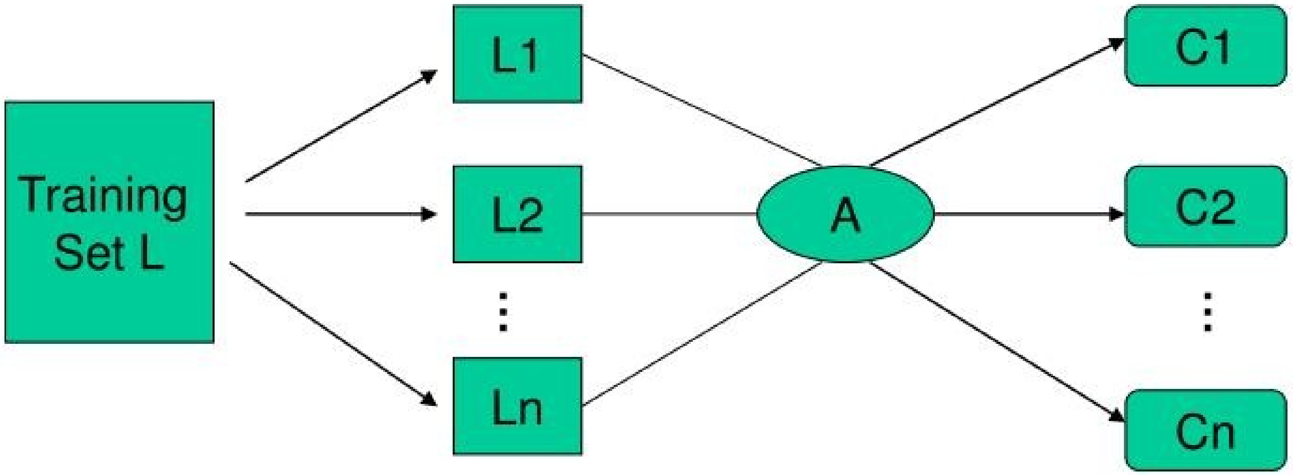


P: parameters for the learning algorithm

Different types of ensemble learning (3)

Same algorithm, different versions of data set, e.g.

- Bagging: resample training data
- Boosting: Reweight training data
- Decorate: Add additional artificial training data
- RandomSubSpace (random forest): random subsets of features



In WEKA, these are called ***meta-learners***, they take a learning algorithm as an argument (***base learner***) and create a new learning algorithm.

Combining an ensemble of classifiers (1)

Voting:

- Classifiers are combined in a static way
- Each base-level classifier gives a (weighted) vote for its prediction
- Plurality vote: each base-classifier predict a class
- Class distribution vote: each predict a probability distribution
  - $p_C(x) = \Sigma_{C \in \mathcal{C}} p_C(x) / | \mathcal{C} |$

Combining an ensemble of classifiers (2)

Stacking: a stack of classifiers

- Classifiers are combined in a dynamically
- A machine learning method is used to learn how to combine the prediction of the base-level classifiers.
- Top level classifier is used to obtain the final prediction from the predictions of the base-level classifiers

Combining an ensemble of classifiers (3)

Cascading:

- Combine classifiers iteratively.
- At each iteration, training data set is extended with the prediction obtained in the previous iteration

Bagging

Create ensembles by "*bootstrap aggregation*", i.e., repeatedly randomly resampling the training data (Brieman, 1996).

Bootstrap: draw N items from X with replacement

Each ***bootstrap sample*** will on average contain 63.2% of the unique training examples, the rest are replicates.

Bagging

- Train *M* learners on *M* bootstrap samples
- Combine outputs by voting (e.g., majority vote)

Decreases error by decreasing the variance in the results due to ***unstable learners***, algorithms (like decision trees and neural networks) whose output can change dramatically when the training data is slightly changed.

Bagging - Aggregate Bootstrapping

Given a standard training set D of size n

For i = 1 .. M

– Draw a sample of size $n^* \le n$ from D uniformly and with replacement

– Learn classifier C_i

Final classifier is a vote of C_1 .. C_M

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).
- Key Insights:
  - Instead of sampling (as in bagging) re-weigh examples!
  - Examples are given weights. At each iteration, a new hypothesis is learned (weak learner) and the examples are reweighted to focus the system on examples that the most recently learned classifier got wrong.
  - Final classification based on weighted vote of weak classifiers

Construct Weak Classifiers

Using Different Data Distribution

- Start with uniform weighting
- During each step of learning
  - Increase weights of the examples which are not correctly learned by the weak learner
  - Decrease weights of the examples which are correctly learned by the weak learner

Idea

- Focus on difficult examples which are not correctly classified in the previous steps
- Intuitive justification: models should be experts that complement each other

Combine Weak Classifiers

Weighted Voting

- Construct strong classifier by weighted voting of the weak classifiers

Idea

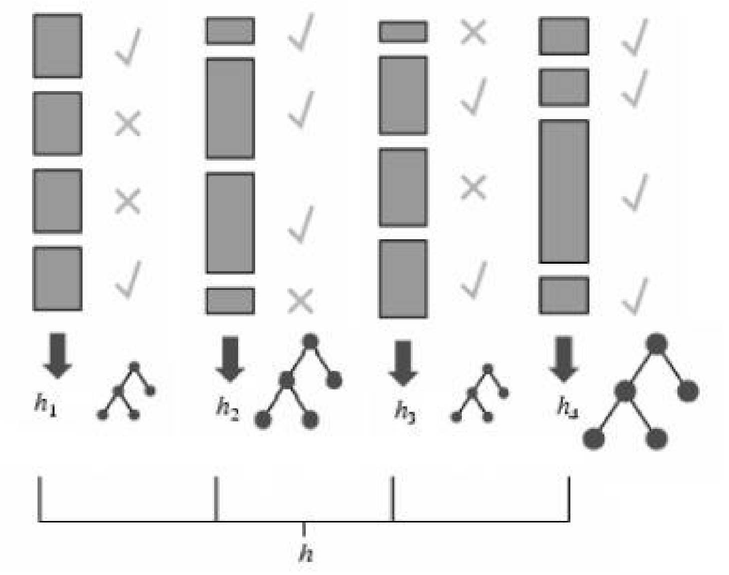
- Better weak classifier gets a larger weight
- Iteratively add weak classifiers
    - Increase accuracy of the combined classifier through minimization of a cost function

Adaptive Boosting

Each rectangle corresponds to an example, with weight proportional to its height.

Crosses correspond to misclassified examples.

Size of decision tree indicates the weight of that hypothesis in the final ensemble.



AdaBoost Pseudocode

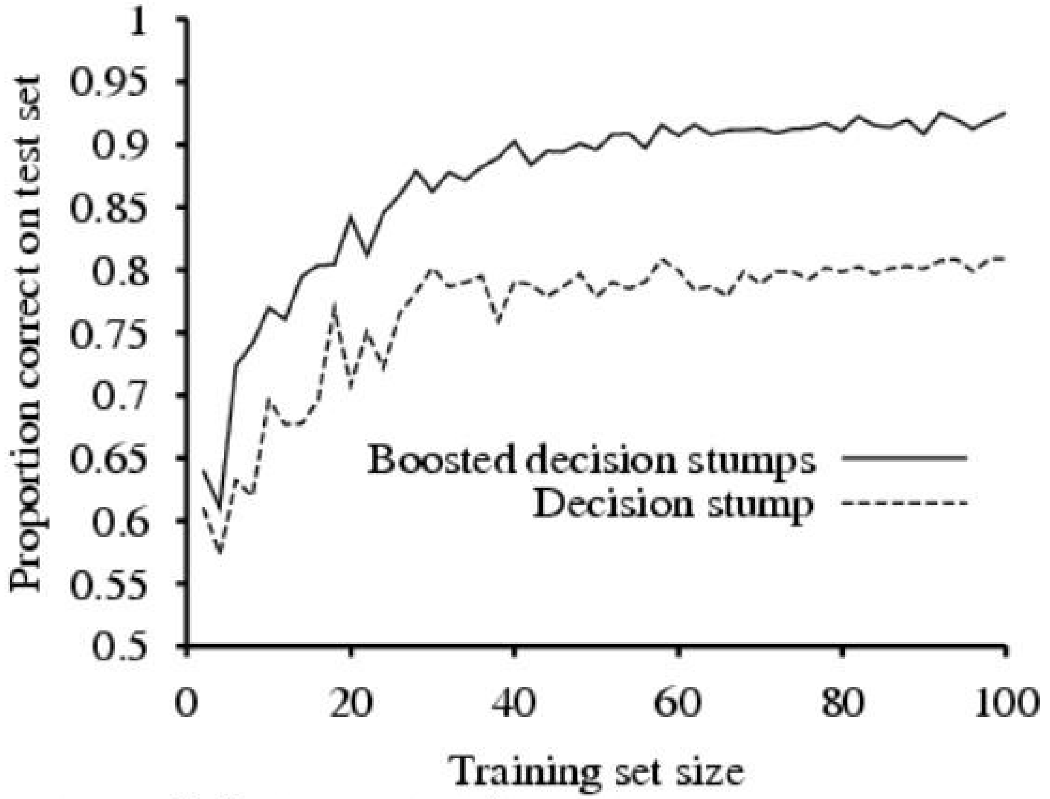
```
TrainAdaBoost(D, BaseLearn)
 For each example $d_i$ in $D$ let its weight $w_i$=1/|$D$|
 Let $H$ be an empty set of hypotheses
 For $t$ from 1 to $T$ do:
     Learn a hypothesis, $h_t$, from the weighted examples: $h_t$=BaseLearn($D$)
     Add $h_t$ to $H$
     Calculate the error, $\varepsilon_t$, 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 $\beta_t$
     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/ $\beta_t$)
    Return the class with the highest weighted vote total.
```

Learning with Weighted Examples

- Generic approach: replicate examples in the training set proportional to 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
- For decision trees, for calculating information gain, when counting example i, simply increment the corresponding count by w_i rather than by 1.
- Can apply boosting without weights
  - resample with probability determined by weights
  - disadvantage: not all instances are used
  - advantage: if error > 0.5, can resample again

Restaurant Data



Decision stump: decision trees with just one test at the root.

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.
- Bagging is easily parallelized, Boosting is not.

Random Forest

- Leo Breiman, *Random Forests*, Machine Learning, 45, 5-32, 2001
- Motivation: reduce error correlation between classifiers
- Main idea: build a larger number of un-pruned decision trees
- Key: using a random selection of features to split on at each node

How Random Forest Work

- Each tree is grown on a bootstrap sample of the training set of **N** cases.
- A number **m** is specified much smaller than the total number of variables **M** (e.g. m = sqrt(M)).
- At each node, m variables are selected at random out of the M.
- The split used is the best split on these **m** variables.
- Final classification is done by majority vote across trees.

Source: Brieman and Cutler

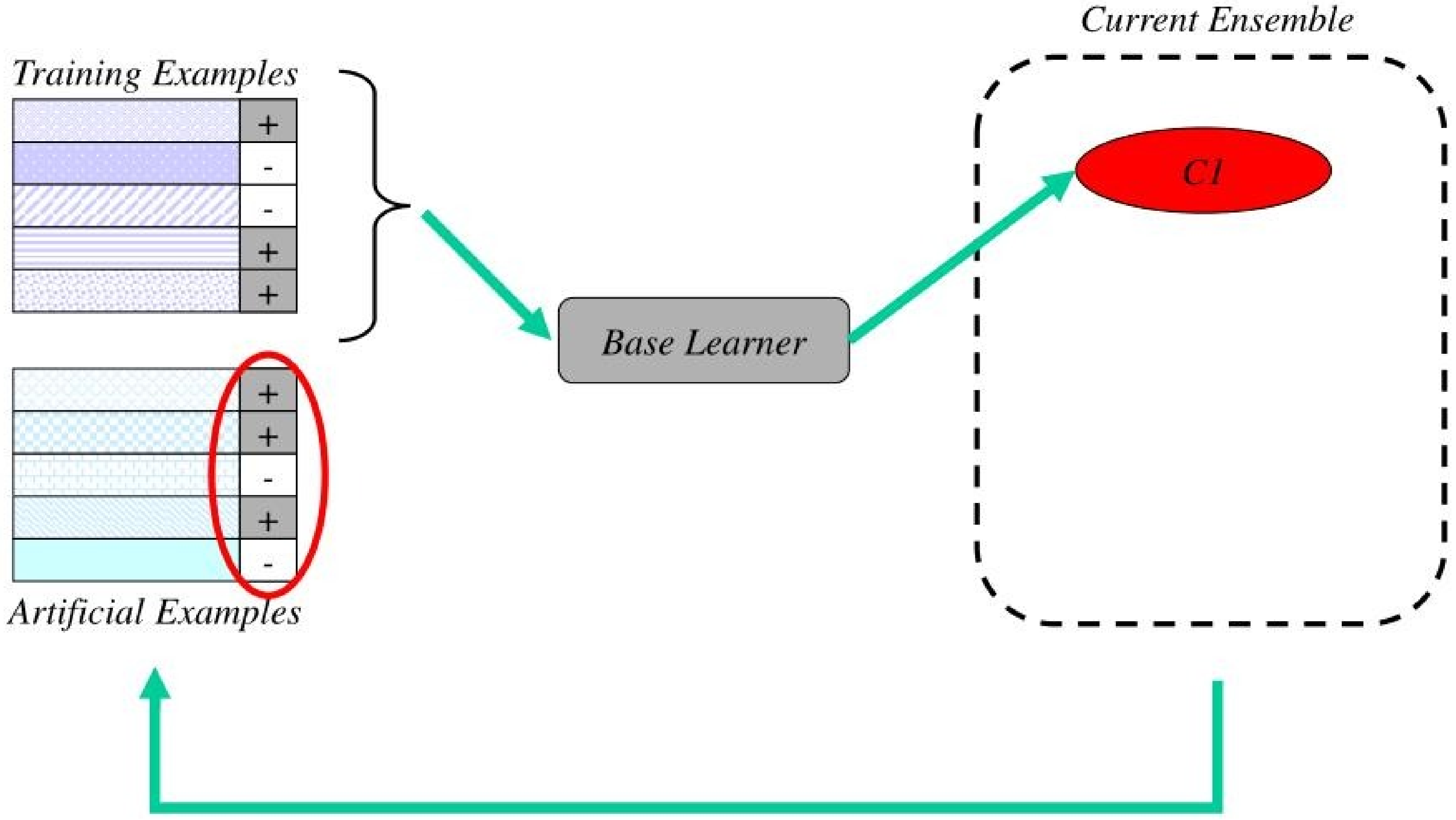
Advantages of random forest

- Error rates compare favorably to Adaboost
- More robust with respect to noise.
- More efficient on large data
- Provides an estimation of the importance of features in determining classification
- More info at: http://stat-www.berkeley.edu/users/breiman/RandomForests/cc_home.htm

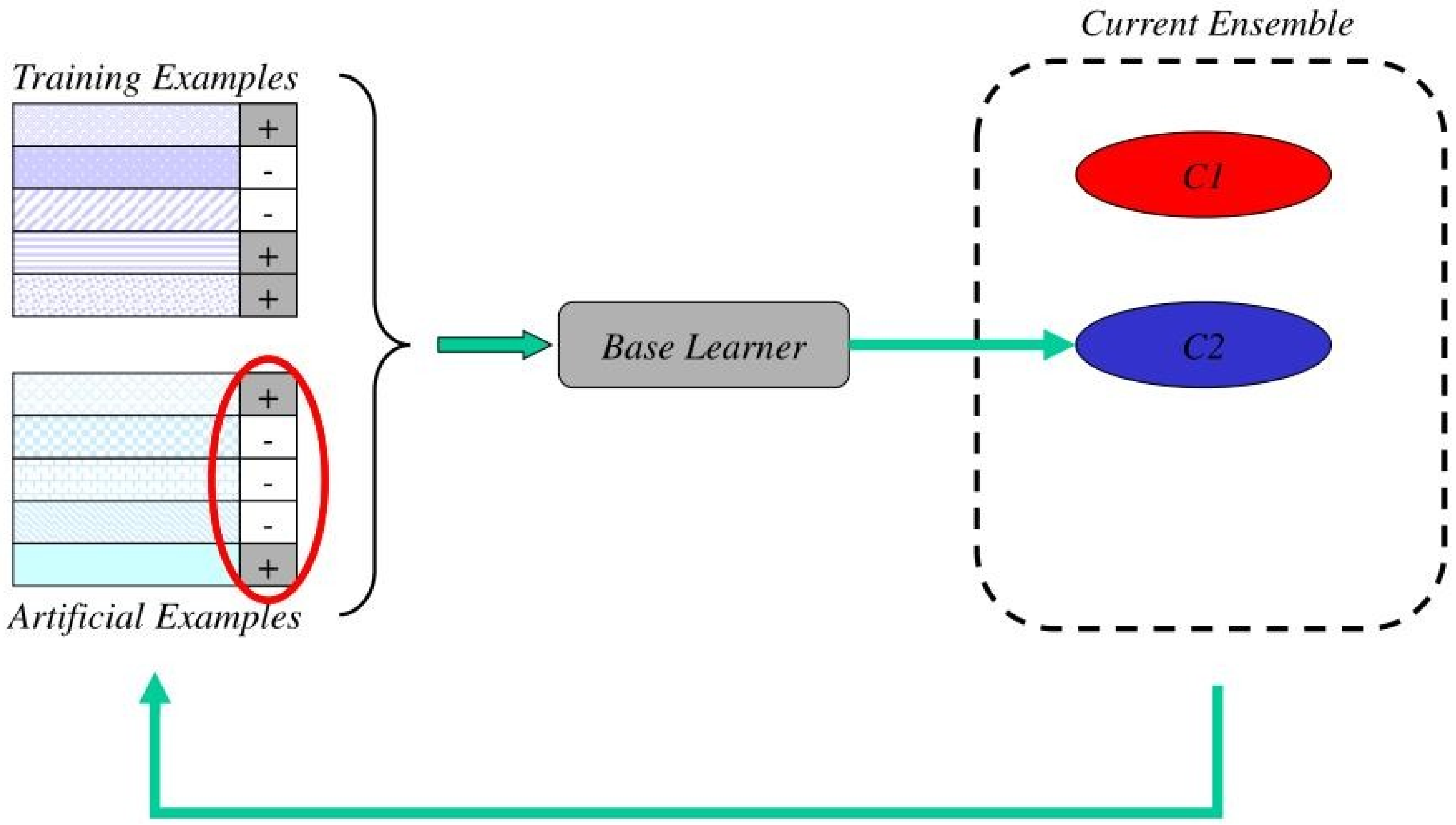
DECORATE (Melville & Mooney, 2003)

- 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.

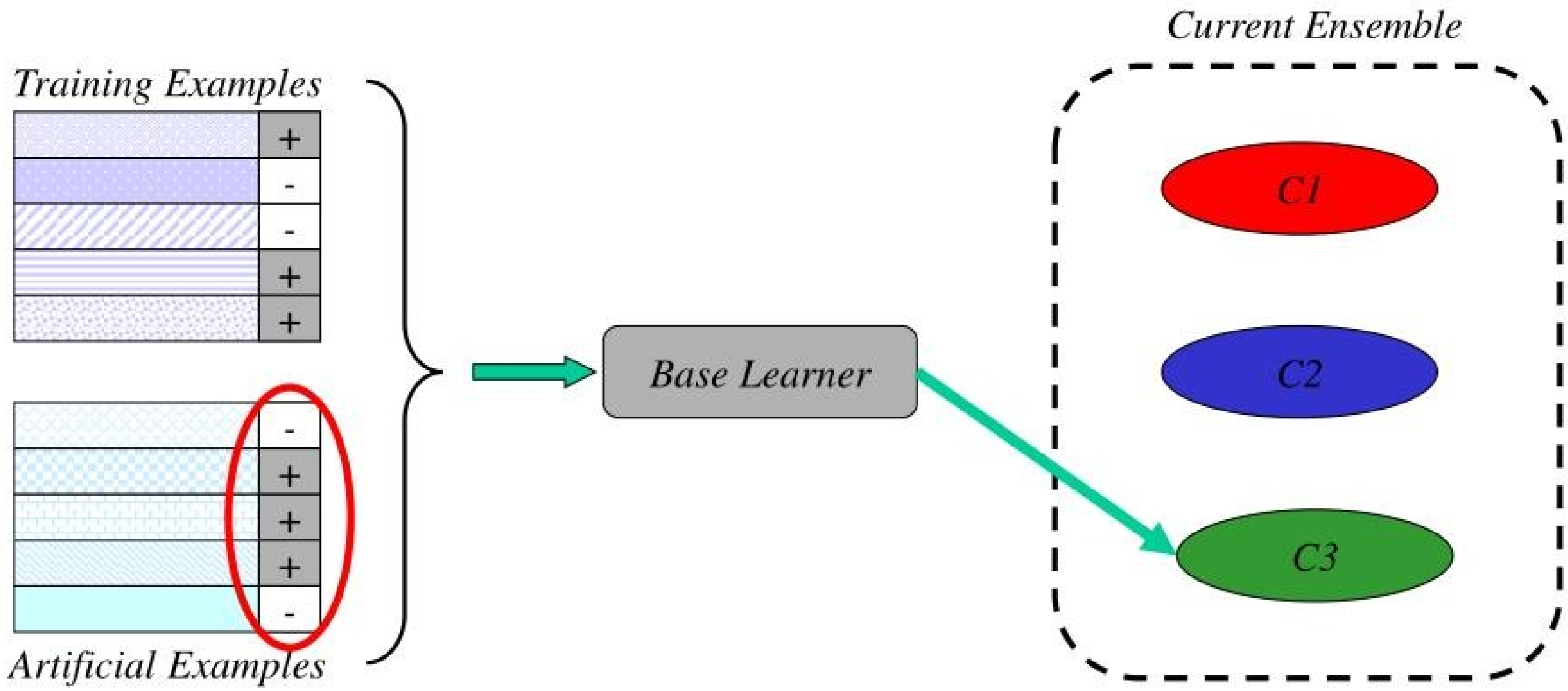
Overview of DECORATE



Overview of DECORATE



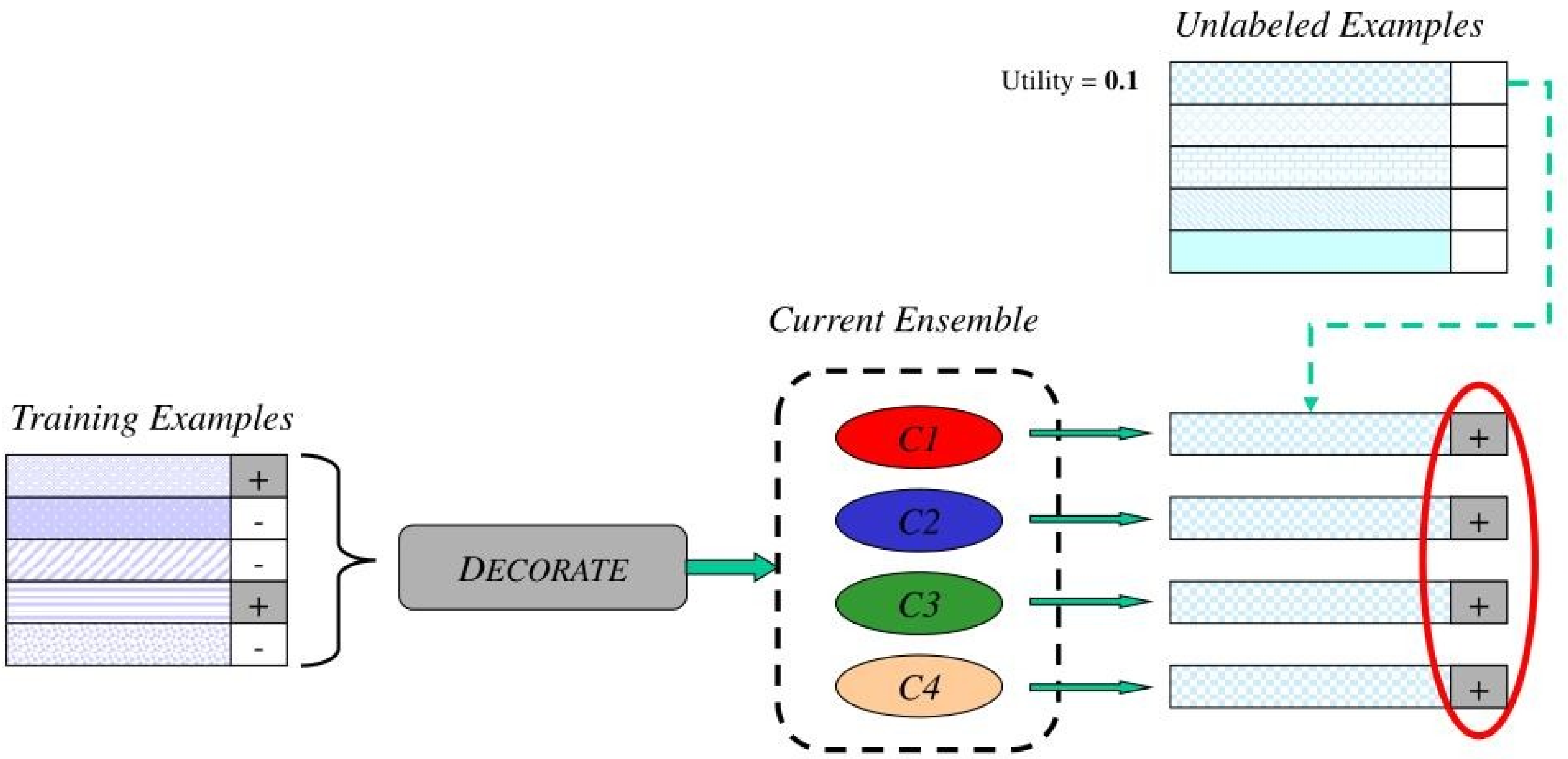
Overview of DECORATE



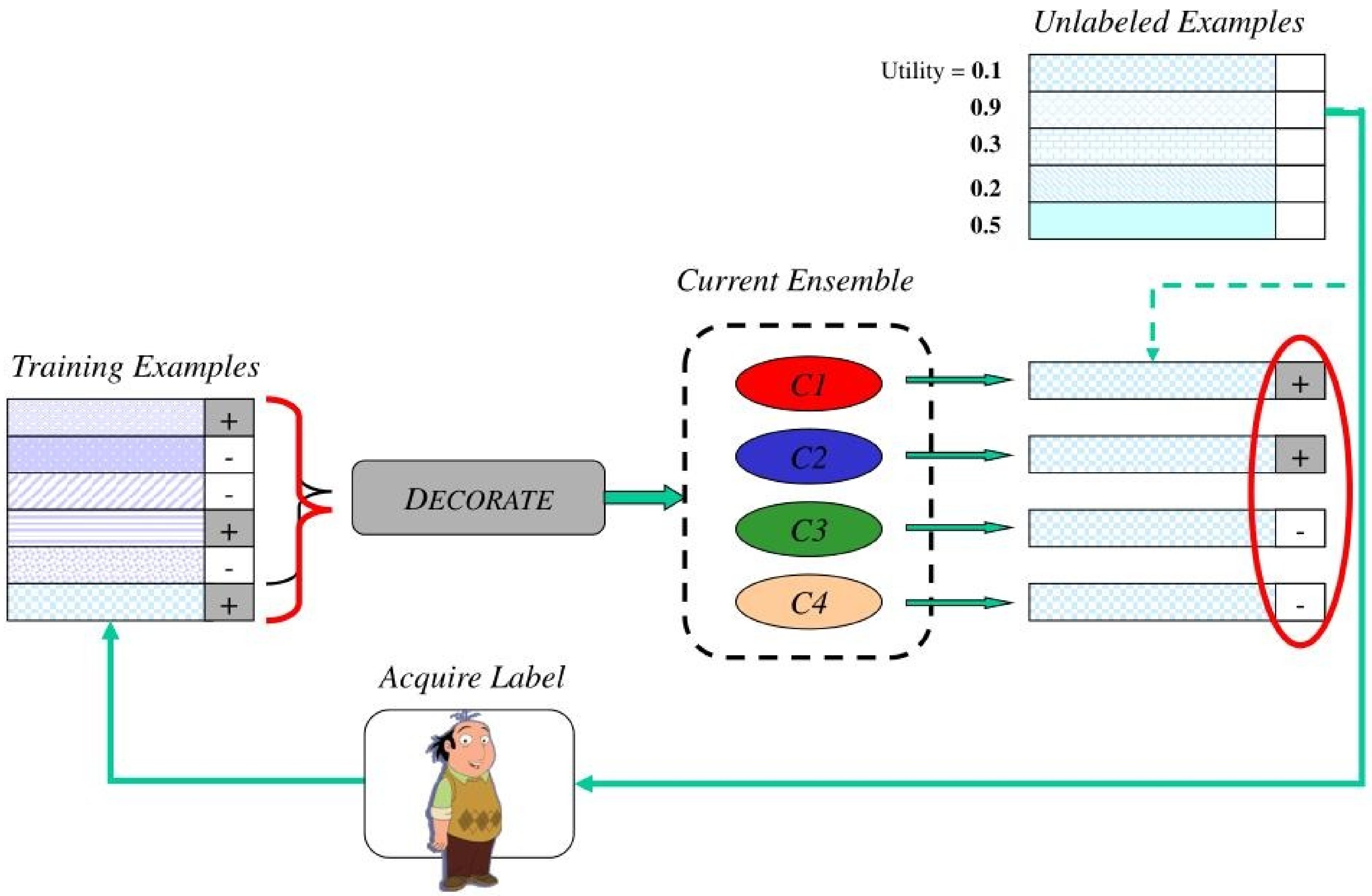
Ensembles and Active Learning

- 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

Active-DECORATE



Active-DECORATE



Meta decision trees (MDT)

Combining Classifiers with Meta Decision Trees, Ljupco Todorovski and Saso Dzeroski, *Machine Learning*, 50, 223-249, 2003

- A type of stacking, usually uses different algorithms for different base-level classifiers
- A MDT is a decision tree whose leaf leads to a prediction of which base-level classifier should be used
- Meta-level attributes could be something calculated from the prediction of the base-level classifiers, or could be just the set of base-level attributes.
- Useful when data include instances from heterogeneous sub-domains

An example

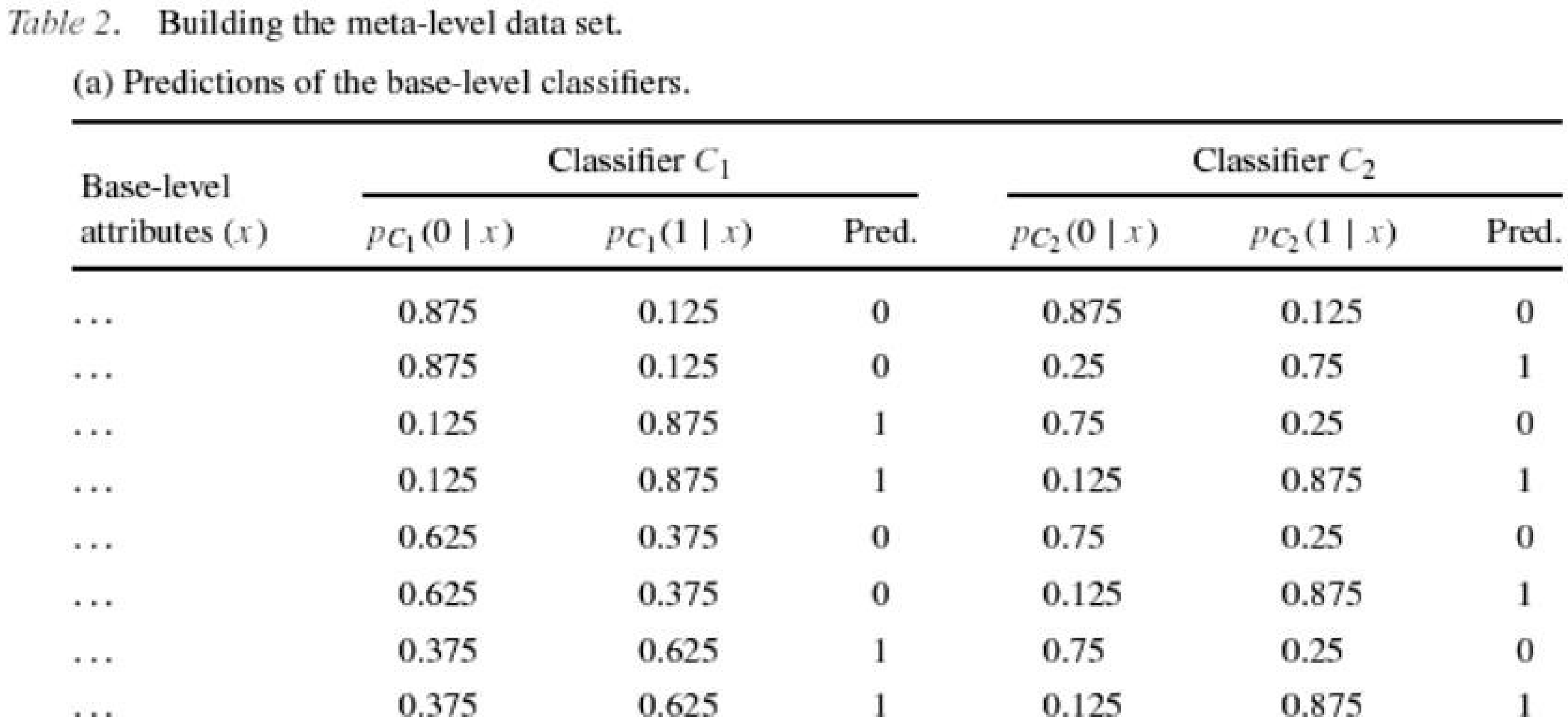


Table 2. Building the meta-level data set.

(a) Predictions of the base-level classifiers.

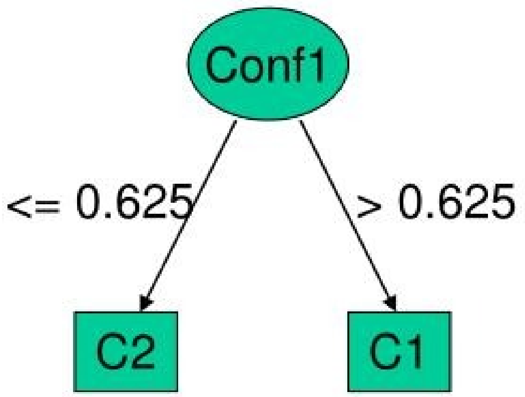
Base-level attributes (x)	Classifier C_1			Classifier C_2		
	$p_{C_1}(0 \mid x)$	$p_{C_1}(1 \mid x)$	Pred.	$p_{C_2}(0 \mid x)$	$p_{C_2}(1 \mid x)$	Pred.
...	0.875	0.125	0	0.875	0.125	0
...	0.875	0.125	0	0.25	0.75	1
...	0.125	0.875	1	0.75	0.25	0
...	0.125	0.875	1	0.125	0.875	1
...	0.625	0.375	0	0.75	0.25	0
...	0.625	0.375	0	0.125	0.875	1
...	0.375	0.625	1	0.75	0.25	0
...	0.375	0.625	1	0.125	0.875	1

Building the meta-level dataset

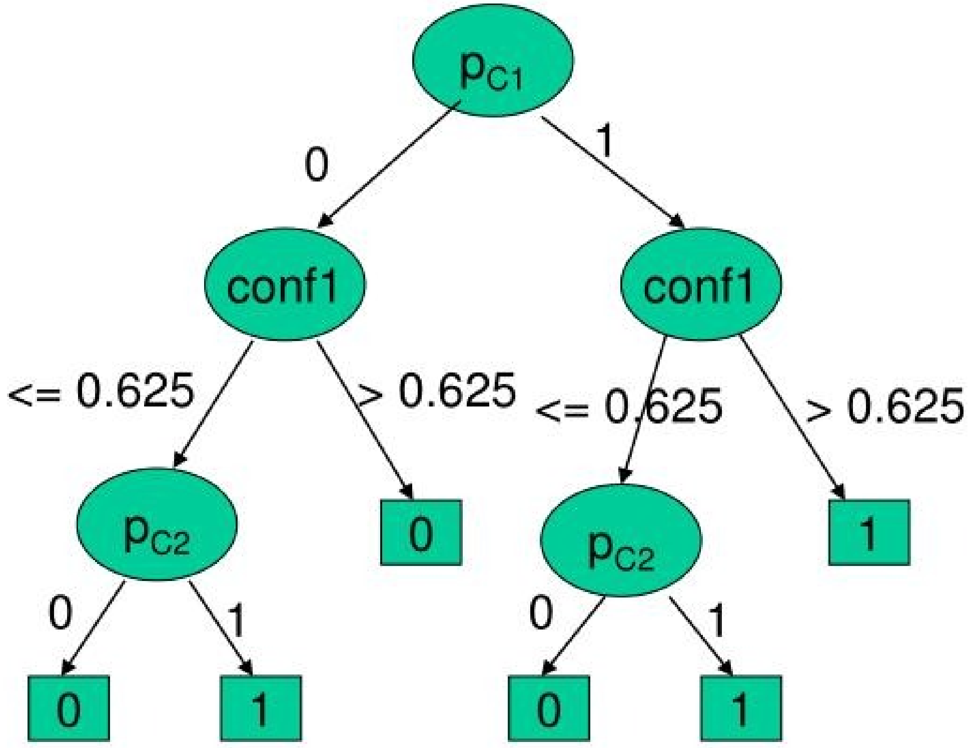
(b) The meta-level data set M.

$Conf_1$	C_1	$Conf_2$	C_2	True class
0.875	0	0.875	0	0
0.875	0	0.75	1	0
0.875	1	0.75	0	1
0.875	1	0.875	1	1
0.625	0	0.75	0	0
0.625	0	0.875	1	1
0.625	1	0.75	0	0
0.625	1	0.875	1	1

Stacking with MDT vs. ODT



(a) Stacking of C1 and C2 with MDT



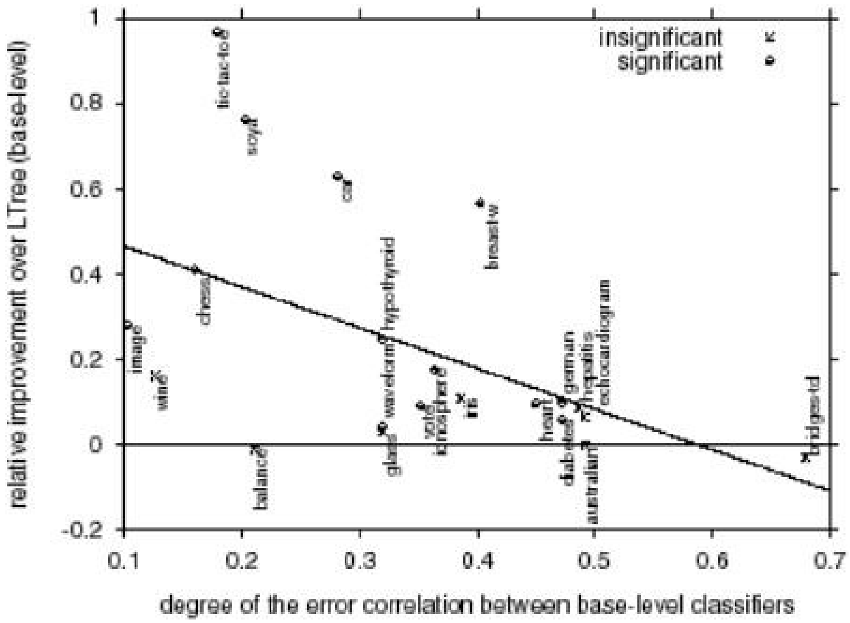
(b) Stacking of C1 and C2 with ODT

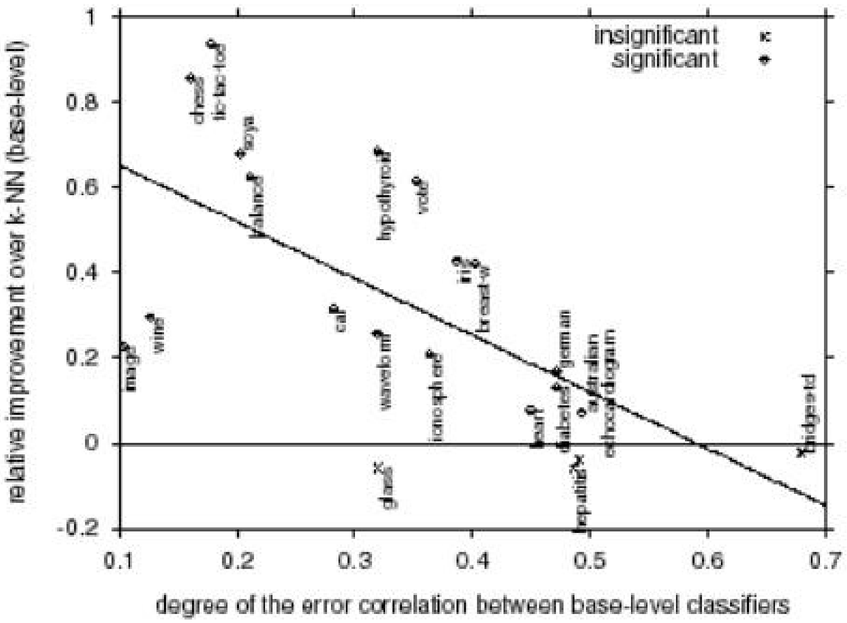
- ODT uses class values predicted by base-level classifiers as attributes. (e.g., the test pc1 = 0), while this is not used by MDT
- ODT predicts the class value in its leaf nodes, while MDT predicts the base-level classifiers to be used

Performance

Meta-level algorithm	Ave. rel. acc. imp. (in %)	Wins/losses	Ave. tree size
C-C4.5	−4.09	15/2	46.01
C-AC4.5	7.90	12/1	95.48
MDT-CDP	0.00	0/0	2.79
MDT-BLA	14.47	8/0	66.94
MDT-CDP+BLA	13.91	8/0	73.73
P-VOTE	22.33	10/3	NA
CD-VOTE	19.64	10/5	NA
Select-Best	7.81	5/2	1
SCANN	13.95	9/6	NA
Boosting	9.36	13/6	NA
Bagging	22.26	10/4	NA

Relative Accuracy Improvement vs. Classifier Diversity





Netflix

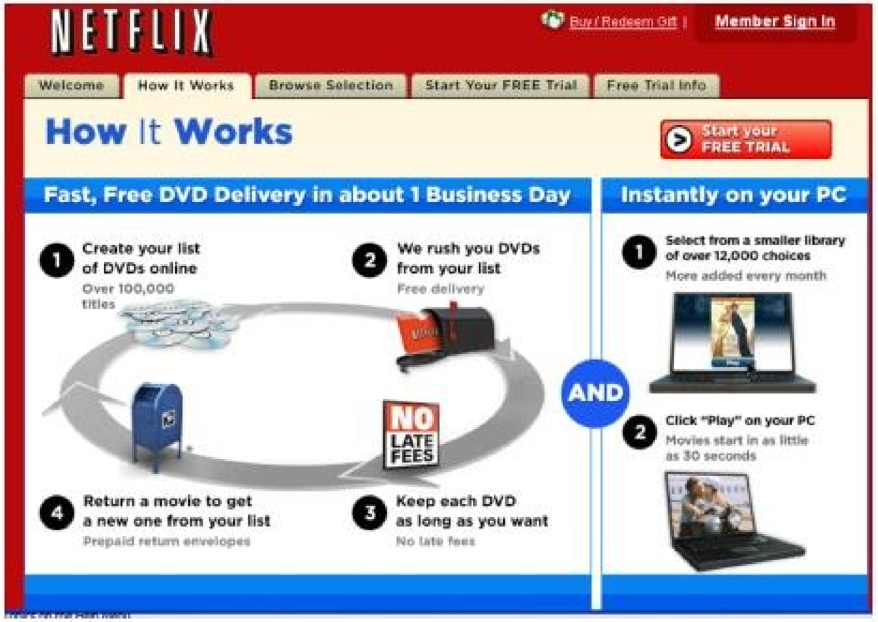
Netflix

NETFLIX
Buy / Redeem Gift | Member Sign In
Welcome
How It Works
Browse Selection
Start Your FREE Trial
Free Trial Info
Give Netflix for the holidays
Give
Plans from only $4.99 a month
Start Now

The Best Way to Rent Movies

FREE TRIAL (Offer Details)

- You'll get free shipping both ways
- Watch classics to new releases to TV series
- Cancel anytime



How It Works

Fast, Free DVD Delivery in about 1 Business Day

1. Create your list of DVDs online — Over 100,000 titles
2. We rush you DVDs from your list — Free delivery
3. Keep each DVD as long as you want — No late fees
4. Return a movie to get a new one from your list — Prepaid return envelopes

Instantly on your PC

1. Select from a smaller library of over 12,000 choices — More added every month
2. Click "Play" on your PC — Movies start in as little as 30 seconds

Users rate movies (1,2,3,4,5 stars);
Netflix makes suggestions to users based on previous rated movies.

Netflix Prize

http://www.netflixprize.com/index

Since October 2006

Welcome!

The Netflix Prize seeks to substantially improve the accuracy of predictions about how much someone is going to love a movie based on their movie preferences. Improve it enough and you win one (or more) Prizes. Winning the Netflix Prize improves our ability to connect people to the movies they love.

Read the Rules to see what is required to win the Prizes. If you are interested in joining the quest, you should register a team.

You should also read the frequently-asked questions about the Prize. And check out how various teams are doing on the Leaderboard.

Good luck and thanks for helping!

"The Netflix Prize seeks to substantially improve the accuracy of predictions about how much someone is going to love a movie based on their movie preferences. Improve it enough and you win one (or more) Prizes. Winning the Netflix Prize improves our ability to connect people to the movies they love."

Netflix Prize

http://www.netflixprize.com/index

Since October 2006

Supervised learning task

– Training data is a set of users and ratin
users have given to movies.

– Construct a classifier that given a user
correctly classifies that movie as either

The Netflix Prize seeks to substantially improve the accuracy of predictions about how much someone is going to love a movie based on their movie preferences. Improve it enough and you win one (or more) Prizes. Winning the Netflix Prize improves our ability to connect people to the movies they love.

Read the Rules to see what is required to win the Prizes. If you are interested in joining the quest, you should register a team.

You should also read the frequently-asked questions about the Prize. And check out how various teams are doing on the Leaderboard.

Good luck and thanks for helping!

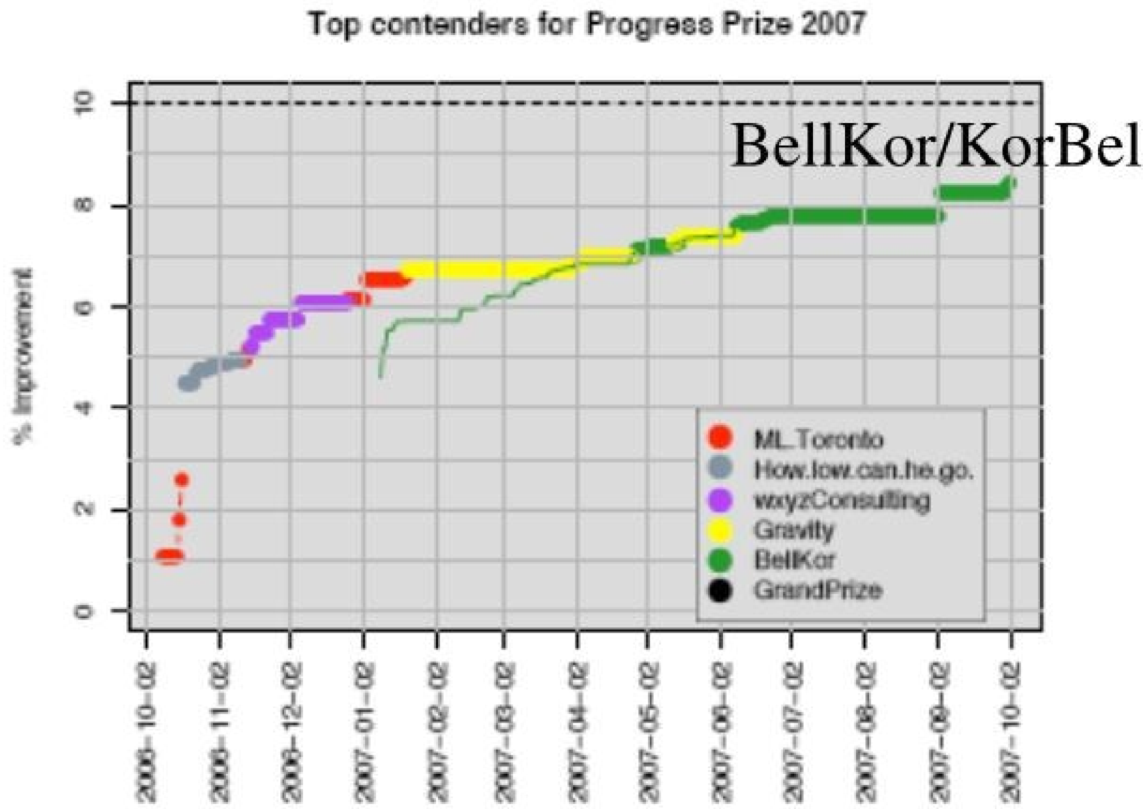
$1 million prize for a 10% improvement over Netflix's current movie recommender/classifier (MSE = 0.9514)

BellKor / KorBell

Scores of the leading team for the first 12 months of the Netflix Prize.

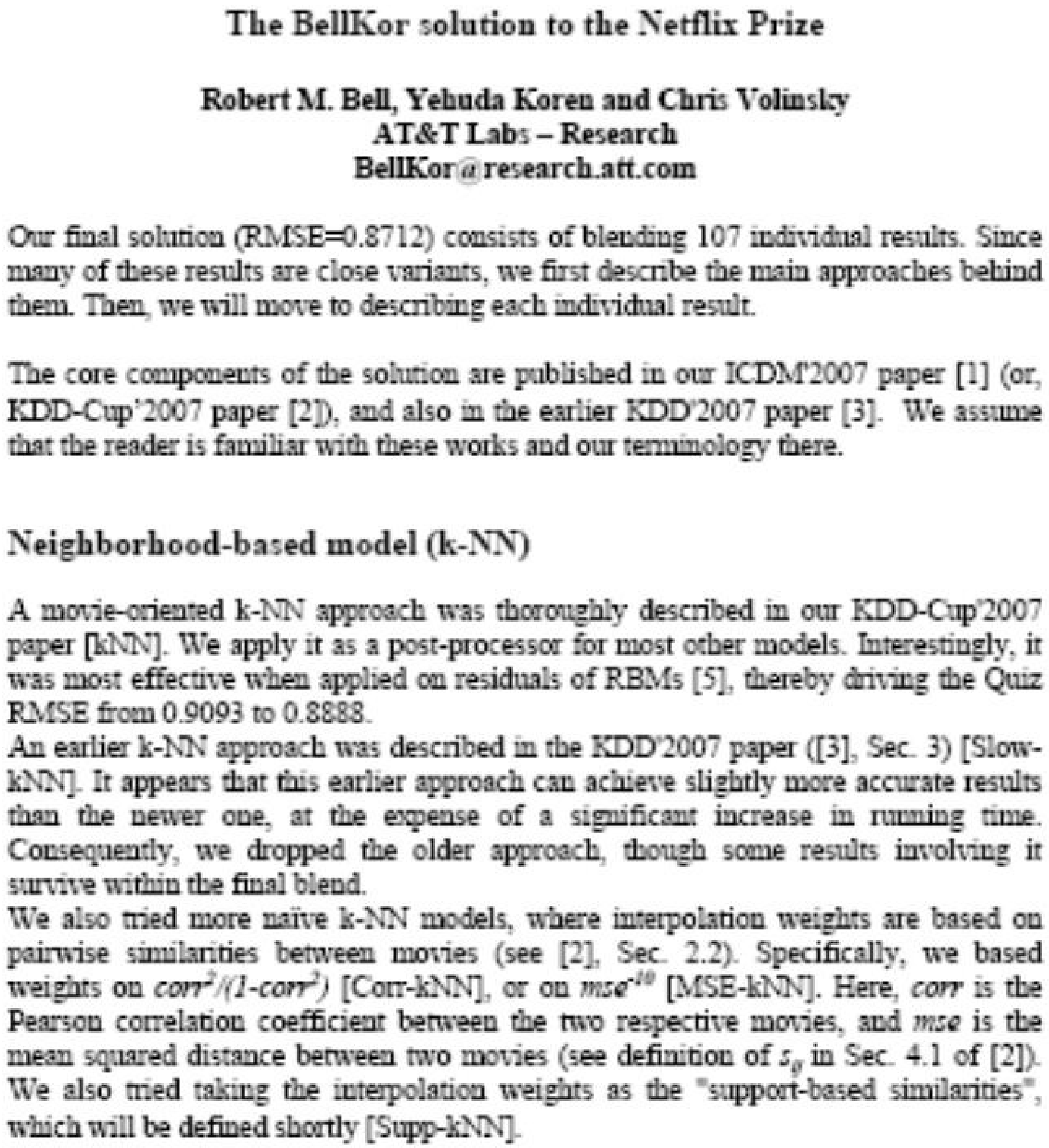
Colors indicate when a given team had the lead. The % improvement is over Netflix' Cinematch algorithm.

The million dollar Grand Prize level is shown as a dotted line at 10% improvement.



from http://www.research.att.com/~volinsky/netflix/

“Our final solution (RMSE=0.8712) consists of blending 107 individual results. “



The BellKor solution to the Netflix Prize

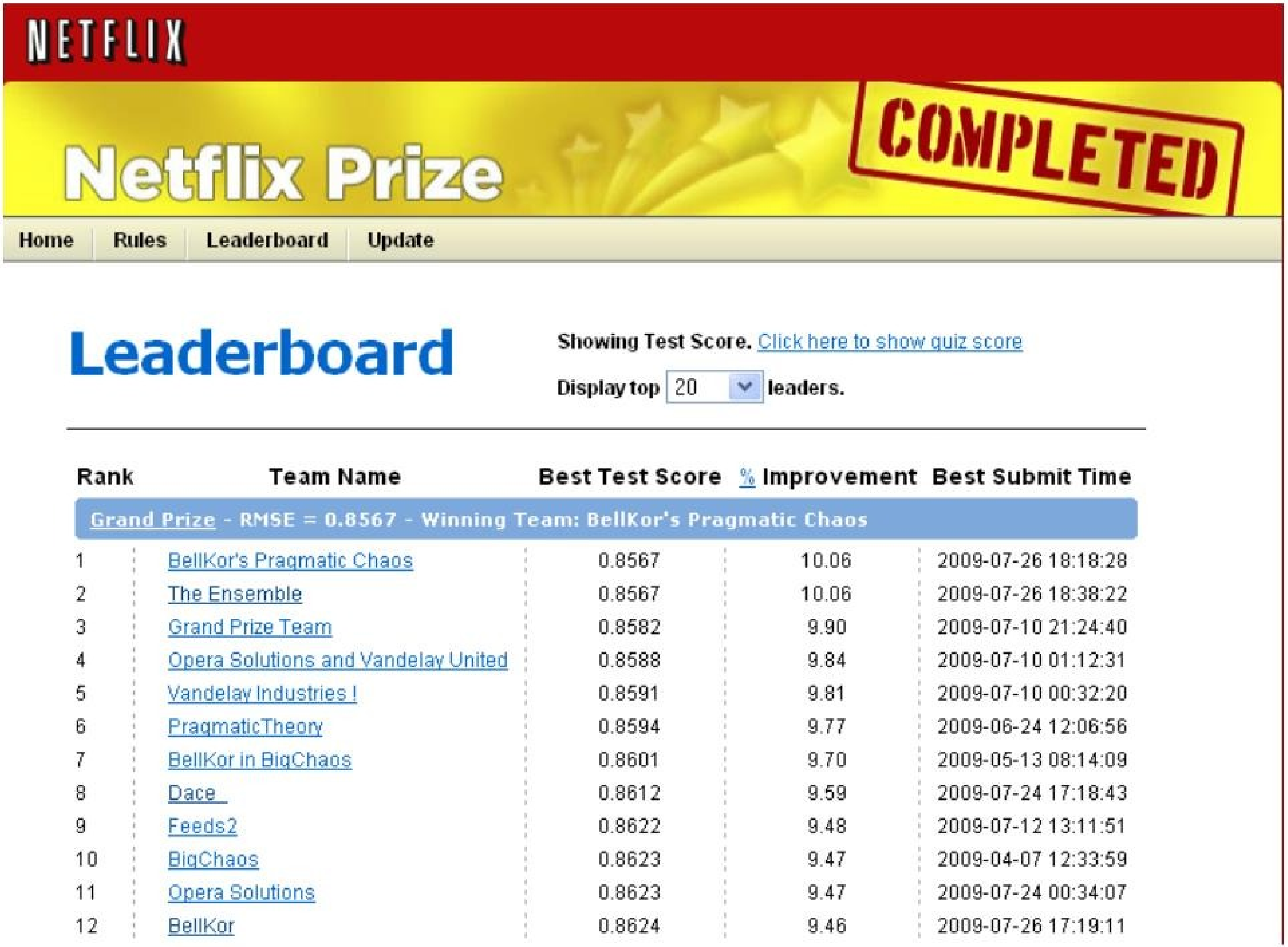
Robert M. Bell, Yehuda Koren and Chris Volinsky
AT&T Labs – Research
BellKor@research.att.com

Our final solution (RMSE=0.8712) consists of blending 107 individual results. Since many of these results are close variants, we first describe the main approaches behind them. Then, we will move to describing each individual result.

The core components of the solution are published in our ICDM'2007 paper [1] (or, KDD-Cup'2007 paper [2]), and also in the earlier KDD'2007 paper [3]. We assume that the reader is familiar with these works and our terminology there.

Neighborhood-based model (k-NN)

A movie-oriented k-NN approach was thoroughly described in our KDD-Cup'2007 paper [kNN]. We apply it as a post-processor for most other models. Interestingly, it was most effective when applied on residuals of RBMs [5], thereby driving the Quiz RMSE from 0.9093 to 0.8888.
An earlier k-NN approach was described in the KDD'2007 paper ([3], Sec. 3) [Slow-kNN]. It appears that this earlier approach can achieve slightly more accurate results than the newer one, at the expense of a significant increase in running time. Consequently, we dropped the older approach, though some results involving it survive within the final blend.
We also tried more naïve k-NN models, where interpolation weights are based on pairwise similarities between movies (see [2], Sec. 2.2). Specifically, we based weights on $corr^2/(1-corr^2)$ [Corr-kNN], or on mse^{-10} [MSE-kNN]. Here, *corr* is the Pearson correlation coefficient between the two respective movies, and *mse* is the mean squared distance between two movies (see definition of s_{ij} in Sec. 4.1 of [2]). We also tried taking the interpolation weights as the "support-based similarities", which will be defined shortly [Supp-kNN].



Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11