Bagging and Boosting

Paul Romanczyk & Wenbo Wang
par4249@rit.edu wxw4213@rit.edu

Oct. 14, 2010

Outline

Outline
Introduction
Bagging and Boosting: the Basic Idea
Bagging
Algorithm Review
Theoretical Analysis
Variants of Bagging
Boosting
Overveiw
Boosting Examples
Example
References
Questions

Bagging and Boosting: the Basic Idea

- Bagging
  - Sample (uniformly) with replacement from the original training set
  - Use unstable base classifier to develop the classifier ensemble iteratively
  - Final decision is based on voting
- Boosting
  - Evolve the probability distribution of classifier ensemble to minimize the loss
  - The classifier in the ensemble is built on a training set sampled from the entire training set with updated distribution
  - Expend the classifier ensemble incrementally

Algorithm Review [Kuncheva, 2004]

- **Training Phase**
  - 1. Initialize the parameters
    - $\mathcal{D} = \emptyset$, the ensemble
    - L, the number of classifiers to train
  - 2. For $k = 1, ..., L$
    - Take a bootstrap sample S_k from $\mathbf{Z}$
    - Build a classifier D_k using S_k as the training set
    - Add the classifier to the current ensemble, $\mathcal{D} = \mathcal{D} \cup D_k$
  - 3. Return $\mathcal{D}$
- **Classification Phase**
  - 4. Run $D_1, ..., D_L$ on the input *mathbfx*
  - 5. The class with the maximum number of votes is chosen as the label for *mathbfx*

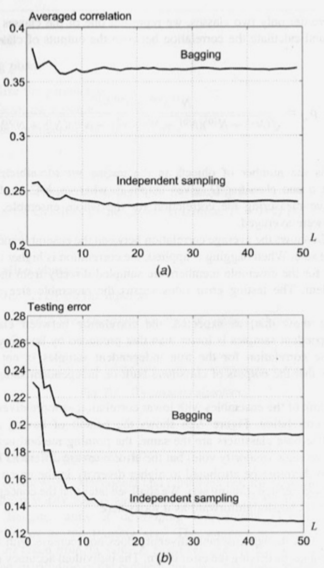
Technique Highlight

- The bootstrap sampling with replacement is drawn from the training set **Z** with the same uniform distribution
- Bagging is a linear combination of classifiers derived from a single base classifier:
  - Majority voting (hard-labeling in the case of binary classification)
  - Soft-combination with weighted output (soft-labeling in the case of binary classification)

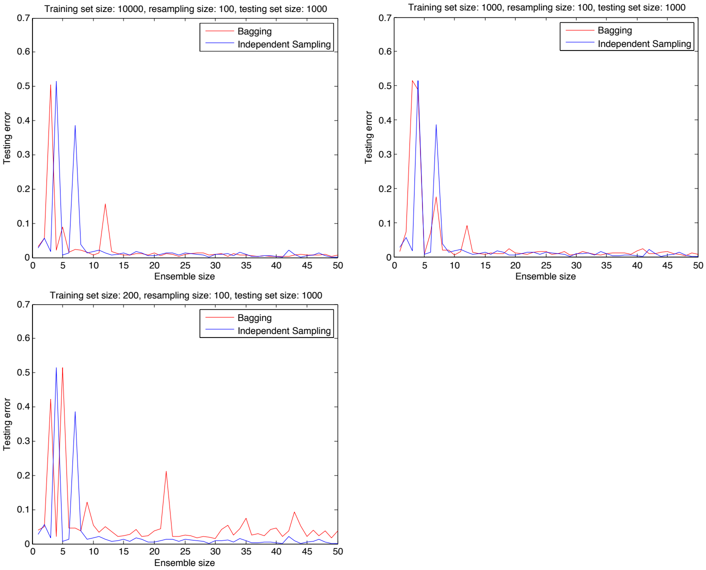
Bootstrap replicates

- Ideal (Independent) Sampling:
  - Build the sub-training set with random sample of the true sample distribution
  - Develop independent classifier
- Idea Bagging [Fumera et al., 2008]
  - Classifier output is the expectation of random bootstrap replicate of **Z**
- Real Bagging
  - A finite approximation of idea bagging

Classifier Correlation [Kuncheva, 2004]

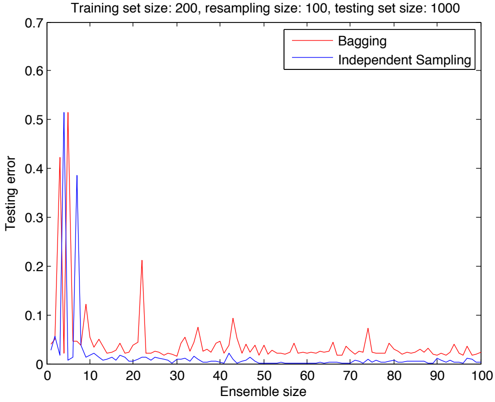


Empirical Analysis of Classifier Correlation (on Check-Board Data)



Empirical Analysis of Classifier Correlation (Cont')

- When we keep increasing the size of 200 training-set ensemble:



Interpretation by Bias-Variance Decomposition [Fumera et al., 2008]

- Average error: $E = E^2_{bayes} + E^2_{bias} + V$
- Bagging reduce the variance by increasing the ensemble size
  - $E_{add} = E^2_{bias} + V = E_T\{E^2(x; t_B) + \frac{1}{m} V(x; t_B)\}$

Random Forest

- Training: build a collection of tree-classifiers, each tree grown with a random vector $\Theta_k, k = 1, ..., L$.
- Decision: Major vote
- Random vector (i.i.d.) Θ_k include:
  - Randomly sample the feature set
  - Randomly sample the training set
  - Randomly varying some parameters

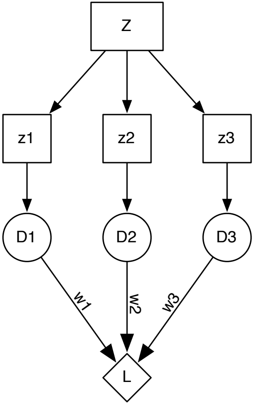
Pasting Small Votes

- Aiming at massive data set
- Training: classifiers are trained on random small sub-set of the training set (called bite)
  - RVote: sampling follows the same distribution
  - IVote.a: new sampling is based on test error of the old ensemble (out-of-bag estimate)
  - IVote.b: use separate validation set
- Decision: Major vote
- Random vector (i.i.d.) Θ_k include:
  - Randomly sample the feature set
  - Randomly sample the training set
  - Randomly varying some parameters

Boosting

- Run multiple classifiers
- Weight the classifiers by how well they perform.
- Unstable classifiers are ideally suited to boosting algorithms that subsample the training data.

Boosting



HEDGE(β)

Given:

- $\mathcal{D} = \{D_1, \ldots, D_L\}$
- $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$

1. Initialize the parameters
   - Pick $\beta \in [0, 1]$
   - Set weights $\mathbf{w}^1 = [w_1, \ldots, w_L]$, $w_i^1 \in [0, 1]$, $\sum_{i=1}^{N} w_i^1 = 1$ (Usually $w_i^1 = \frac{1}{L}$)
   - Set cumulative loss $\Lambda = 0$
   - Set individual loss $\lambda_i = 0,\ i = 1, \ldots, L$

HEDGE(β)

2. For all $\mathbf{z}_j$, $j = 1, \ldots, N$
   - Calculate the distribution by
     $$p_i^j = \frac{w_i^j}{\sum_{k=1}^{L} w_k^j}, i = 1, \ldots, L$$
   - Find the individual losses: $l_i^j = 1$ if D_i produces a misclassification of $\mathbf{z}_j$ and $l_i^j = 0$ otherwise, $i = 1, \ldots, L$)
   - Update the cumulative loss
     $$\Lambda \leftarrow \Lambda + \sum_{i=1}^{L} p_i^j l_i^j$$
   - Update the individual losses
     $$\lambda_i \leftarrow \lambda_i + l_i^j$$
   - Update the weights
     $$w_i^{j+1} = w_i^j \beta^{l_i^j}$$

HEDGE(β)

3. Calculate and return Λ, λ_i, and p_i^{N+1}, $i = 1, \dots, L$.

AdaBoost

Adaptive **Boost**ing

Training

1. Initialize the parameters
   - Set weights $\mathbf{w}^1 = [w_1, \ldots, w_N], w_j^1 \in [0, 1], \sum_{j=1}^{N} w_j^1 = 1$
   - Initialize the ensemble $\mathcal{D} = \emptyset$
   - Pick the number of classifiers to train, L

AdaBoost

2. For $k = 1, \ldots, L$
   - Take a sample S_k from $\mathbf{Z}$ using distribution $\mathbf{w}^k$
   - Build a classifier D_k using S_k as a training set
   - Calculate the weighted ensemble error at step k by $\epsilon_k = \sum_{j=1}^{N} w_j^k l_k^j$ where $l_j^k = 1$ if D_k produces a misclassification of $\mathbf{z}_j$ and $l_j^k = 0$ otherwise.
   - If $\epsilon_k = 0$ or $\epsilon_k \geq 0.5$, ignore D_k, reinitialize the weights w_j^k to $1/N$ and continue.
   - Else calculate $\beta_k = \frac{\epsilon_k}{1-\epsilon_k}$, $\epsilon_k \in (0, 0.5)$
   - Update the individual weights $w_j^{k+1} = \frac{w_j^k \beta^{(1-l_k^j)}}{\sum_{i=1}^{N} w_j^k \beta^{(1-l_k^j)}}$, $j = 1, \ldots, N$
   - Return $\mathcal{D}$ and $\beta_1, \ldots, \beta_L$

AdaBoost

Classification

3. Calculate the support for class ω_t by

$$\mu_t(\mathbf{x}) = \sum_{D_k(\mathbf{x})=\omega_t} \ln\left(\frac{1}{\beta_k}\right)$$

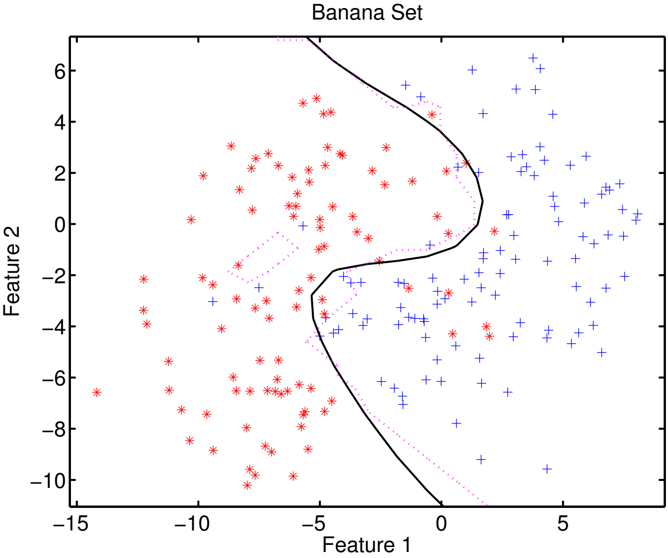
4. The class with the maximum support is chosen as the label for $\mathbf{x}$

Matlab Example

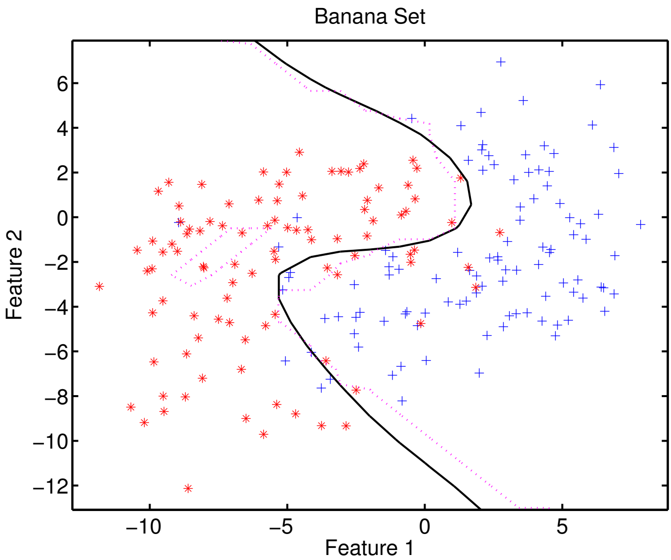
Requires PRtools and Neural Net Toolbox

```
% Generate some data
data = gendatb( 400, 2 );
[ Test, Train ] = gendat( data, 0.5 );

% Train the classifires
w_nn = bpxnc( Train );
w_boost = adaboostc( Train, bpxnc, 4 );

% Classify both data subsets with both trained
% classifieres
nn_train_class = Train * w_nn;
boost_train_class = Train * w_boost;
nn_test_class = Test * w_nn;
boost_test_class = Test * w_boost;
```

Matlab Example



Training set (black line is BPNN, maroon line is boosting)



Testing set (black line is BPNN, maroon line is boosting)

Matlab Example

- Neural Net
- Training Error: 0.071147

True Labels	Estimated Labels 1	2	Totals
1	98	5	103
2	9	87	96
Totals	107	92	199

- Boosting
- Training Error: 0.05623

True Labels	Estimated Labels 1	2	Totals
1	100	3	103
2	8	88	96
Totals	108	91	199

Matlab Example

- Neural Net
- Testing Error: 0.09551

True Labels	Estimated Labels 1	2	Totals
1	97	7	104
2	12	85	97
Totals	109	92	201

- Boosting
- Training Error: 0.14567

True Labels	Estimated Labels 1	2	Totals
1	93	11	104
2	18	79	97
Totals	111	90	201

References

Fumera, G., Roli, F., and Serrau, A. (2008).
A theoretical analysis of bagging as a linear combination of classifiers.
Pattern Analysis and Machine Intelligence, IEEE Transactions on, 30(7):1293 –1299.

Kuncheva, L. (2004).
Combining pattern classifiers: methods and algorithms.
Wiley-Interscience.

Questions