Bagging and Boosting

Amit Srinet
Dave Snyder
Outline

● Bagging
  o Definition
  o Variants
  o Examples
● Boosting
  o Definition
  o Hedge(β)
  o AdaBoost
  o Examples
● Comparison
Bagging

- Bootstrap Model
- Randomly generate $L$ set of cardinality $N$ from the original set $Z$ with replacement.
- Corrects the optimistic bias of R-Method

- "Bootstrap Aggregation"
- Create Bootstrap samples of a training set using sampling with replacement.
- Each bootstrap sample is used to train a different component of base classifier
- Classification is done by plurality voting
Bagging

- Regression is done by averaging
- Works for unstable classifiers
  - Neural Networks
  - Decision Trees
BAGGING

Training phase
1. Initialize the parameters
   • $\mathcal{D} = \emptyset$, the ensemble.
   • $L$, the number of classifiers to train.
2. For $k = 1, \ldots, L$
   • Take a bootstrap sample $S_k$ from $\mathcal{Z}$.
   • 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 $D_1, \ldots, D_L$ on the input $x$.
5. The class with the maximum number of votes is chosen as the label for $x$.

Fig. 7.1  The bagging algorithm.
Example

- PR Tools:

```matlab
>> A = gendatb(500,1);
>> scatterd(A)
>> W1 = baggingc(A,treec,100,[],[]);
>> plotc(W1(:,1:2),'r')
>> W2 = baggingc(A,treec,100,treec,[]);
>> plotc(W2)
```

- Generates 100 trees with default settings - stop based on purity metric, zero pruning
Example

Bagging: Decision Tree

Banana Set

Training data

Banana Set

Decision boundary produced by one tree

Bagging: Decision Tree
Example

Banana Set

Decision boundary produced by a second tree

Decision boundary produced by a third tree

Bagging: Decision Tree
Example

Three trees and final boundary overlaid

Final result from bagging all trees.

Bagging: Decision Tree
Three neural nets generated with default settings [bpxnc]

Final output from bagging 10 neural nets

Bagging: Neural Net
Why does bagging work?

- Main reason for error in learning is due to noise, bias, and variance.
- Noise is error by the target function.
- Bias is where the algorithm cannot learn the target.
- Variance comes from the sampling and how it affects the learning algorithm.
- Does bagging minimize these errors?
  - Yes
- Averaging over bootstrap samples can reduce error from variance especially in case of unstable classifiers.
Bagging

- In fact, Ensemble reduces variance.
- Let $f(x)$ be the target value of $x$ and $h_1$ to $h_n$ be the set of base hypotheses and $h$-average be the prediction of base hypotheses.
- $E(h,x) = (f(x) - h(x))^2$ Squared Error.
**Ensemble Reduces variance**

- Let $f(x)$ be the target value for $x$.
- Let $h_1, \ldots, h_n$ be the base hypotheses.
- Let $h$-avg be the average prediction of $h_1, \ldots, h_n$.
- Let $E(h, x) = (f(x) - h(x))^2$
- Is there any relation between $h$-avg and variance?
  - yes
\[ E(h_{\text{avg}}, x) = \frac{1}{n} \sum_{i=1}^{n} E(h_i, x) \]
\[ \frac{1}{n} \sum_{i=1}^{n} (h_i(x) - h_{\text{avg}}(x))^2 \]

That is squared error of the average prediction equals the average squared error of the base hypotheses minus the variance of the base hypotheses.

Reference – 1-End of the slideshow.
Bagging - Variants

● Random Forests
  ○ A variant of bagging proposed by Breiman
  ○ It’s a general class of ensemble building methods using a decision tree as base classifier.
● Classifier consisting of a collection of tree-structure classifiers.
● Each tree grown with a random vector $V_k$ where $k = 1, \ldots, L$ are independent and statistically distributed.
● Each tree cast a unit vote for the most popular class at input $x$. 
Boosting

- A technique for combining multiple base classifiers whose combined performance is significantly better than that of any of the base classifiers.
- Sequential training of weak learners
  - Each base classifier is trained on data that is weighted based on the performance of the previous classifier
- Each classifier votes to obtain a final outcome
Figure 14.2  Illustration of boosting in which the base learners consist of simple thresholds applied to one or other of the axes. Each figure shows the number \( m \) of base learners trained so far, along with the decision boundary of the most recent base learner (dashed black line) and the combined decision boundary of the ensemble (solid green line). Each data point is depicted by a circle whose radius indicates the weight assigned to that data point when training the most recently added base learner. Thus, for instance, we see that points that are misclassified by the \( m = 1 \) base learner are given greater weight when training the \( m = 2 \) base learner.
Boosting - Hedge(β)

- Boosting follows the model of online algorithm.
- Algorithm allocates weights to a set of strategies and used to predict the outcome of the certain event.
- After each prediction the weights are redistributed.
- Correct strategies receive more weights while the weights of the incorrect strategies are reduced further.
- Relation with Boosting algorithm.
- Strategies corresponds to classifiers in the ensemble and the event will correspond to assigning a label to sample drawn randomly from the input.
**HEGDE (β)**

Given:
- \( \mathcal{D} = \{D_1, \ldots, D_L\} \): the classifier ensemble (\( L \) strategies)
- \( \mathcal{Z} = \{z_1, \ldots, z_N\} \): the data set (\( N \) trials).

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

2. For every \( z_j, j = 1, \ldots, N \),
   - Calculate the distribution by
     \[
     p_i^j = \frac{w_i^j}{\sum_{k=1}^{L} w_k^j}, \quad i = 1, \ldots, L. \tag{7.5}
     \]
   - Find the \( L \) individual losses.
     (\( t_i^j = 1 \) if \( D_i \) misclassifies \( z_j \) and \( t_i^j = 0 \) if \( D_i \) classifies \( z_j \) correctly, \( i = 1, \ldots, L \)).
   - Update the cumulative loss
     \[
     \Lambda \leftarrow \Lambda + \sum_{i=1}^{L} p_i^j t_i^j. \tag{7.6}
     \]
   - Update the individual losses
     \[
     \lambda_i \leftarrow \lambda_i + t_i^j. \tag{7.7}
     \]
   - Update the weights
     \[
     w_i^{j+1} = w_i^j \beta^j. \tag{7.8}
     \]

3. Calculate the return \( \Lambda, \lambda_i, \) and \( p_i^{N+1}, i = 1, \ldots, L \).
Boosting - AdaBoost

- Start with equally weighted data, apply first classifier
- Increase weights on misclassified data, apply second classifier
- Continue emphasizing misclassified data to subsequent classifiers until all classifiers have been trained
ADABOOST.M1

Training phase

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

2. For \( k = 1, \ldots, L \)
   - Take a sample \( S_k \) from \( Z \) using distribution \( w^k \).
   - Build a classifier \( D_k \) using \( S_k \) as the training set.
   - Calculate the weighted ensemble error at step \( k \) by
     \[
     \epsilon_k = \sum_{j=1}^{N} w_j^k t_j^k, \tag{7.11}
     \]
     \( (t_j^k = 1 \text{ if } D_k \text{ misclassifies } z_j \text{ and } t_j^k = 0 \text{ otherwise.}) \)
   - If \( \epsilon_k = 0 \) or \( \epsilon_k \geq 0.5 \), ignore \( D_k \), reinitialize the weights \( w_j^k \) to \( \frac{1}{N} \) and continue.
   - Else, calculate
     \[
     \beta_k = \frac{\epsilon_k}{1 - \epsilon_k}, \text{ where } \epsilon_k \in (0, 0.5), \tag{7.12}
     \]
   - Update the individual weights
     \[
     w_j^{k+1} = \frac{w_j^k \beta_k^{1-t_j^k}}{\sum_{j=1}^{N} w_j^k \beta_k^{1-t_j^k}}, \quad j = 1, \ldots, N. \tag{7.13}
     \]

3. Return \( D \) and \( \beta_1, \ldots, \beta_L \).

Classification phase

4. Calculate the support for class \( \omega_i \) by
   \[
   \mu_i(x) = \sum_{D_i(x) = \omega_i} \ln \left( \frac{1}{\beta_k} \right). \tag{7.14}
   \]

5. The class with the maximum support is chosen as the label for \( x \).

Fig. 7.7 The AdaBoost.M1 algorithm with resampling.
Boosting - AdaBoost

- Training error: Kuncheva 7.2.4
- In practice overfitting rarely occurs (Bishop)
Margin Theory

- Testing error continues to decrease
- Ada-boost brought forward margin theory
- Margin for an object is related to certainty of its classification.
- Positive and large margin – correct classification
- Negative margin - Incorrect Classification
- Very small margin – Uncertainty in classification
- Similar classifier can give different label to an input.
- Margin of object $x$ is calculated using the degree of support.

$$m(x) = \mu_k(x) - \max_{j \neq k} \{\mu_j(x)\}$$

Where

$$\sum_{j=1}^{c} \mu_j(x) = 1$$
Freund and Schapire proved upper bounds on the testing error that depend on the margin.

Let $H$ a finite space of base classifiers. For $\delta > 0$ and $\theta > 0$ with probability at least $1 - \delta$ over the random choice of the training set $Z$, any classifier ensemble $D = \{D_1, \ldots, D_L\} \leq H$ combined by the weighted average satisfies
\[ P(\text{error}) \leq P(\text{training margin}) \leq \theta + O\left( \frac{1}{\sqrt{N}} \left( \frac{\log N \log |H|}{\theta^2} \right) + \log(1/\delta)^{1/2} \right) \]

\[
P(\text{error}) = \text{probability that the ensemble will make an error in labeling } x \text{ drawn randomly from the distribution of the problem}
\]

\[
P(\text{training margin} < \theta) = \text{the probability that the margin for a randomly drawn data point from a randomly drawn training set does not exceed } \theta
\]
Thus the main idea for boosting is to approximate the target by approximating the weight of the function.

These weights can be seen as the min-max strategy of the game.

Thus we can apply the notion of game theory for ada-boost.

This idea has been discussed in the paper of freund and schpaire.
Experiment

PR Tools:

```
>> A = gendatb(500, 1);
>> [W,V,ALF] = adaboostc(A,qdc,20,[],1);
>> scatterd(A)
>> plotc(W)
```

* Uses Quadratic Bayes Normal Classifier with default settings, 20 iterations.
Example

**AdaBoost: QDC**

Each QDC classification boundary (black), Final output (red)

Final output of AdaBoost with 20 QDC classifiers
Experiments

AdaBoost: Decision Tree

AdaBoost using 20 decision trees with default settings

Final output of AdaBoost with 20 decision trees
Experiments

AdaBoost using 20 neural nets [bpXnc] default settings

Final output of AdaBoost with 20 neural nets

AdaBoost: Neural Net
Bagging & Boosting

- Comparing bagging and boosting:

Fig. 7.11  Training and testing error of bagging and AdaBoost for the rotated check-board example.
References