Boosting

IST557 Data Mining: Techniques and Applications
Jessie Li, Penn State University

Materials from Prof. Adams, Prof. Sontag, Prof. Ihler, statistical learning book (Hastie et al.), and machine learning book (Murphy)
Overview

• Ensemble learning methods are meta-algorithms that pool decisions from multiple classifiers

• Bagging and boosting are two frequently-used types of ensembles
Bagging: Subsample

Original Data
Bagging: Subsample

Original Data

Subsampled Data Set 1
Bagging: Subsample

Original Data

Subsampled Data Set 1
Bagging: Subsample

Original Data

Subsampled Data Set 1

Subsampled Data Set 2
Bagging: Subsample

Original Data

Subsampled Data Set 1

Subsampled Data Set 2
Bagging: Subsample
Bagging: Subsample

Original Data

Subsampled Data Set 1
Subsampled Data Set 2
Subsampled Data Set 3
Bagging: Subsample

Original Data

Subsampled Data Set 1
Subsampled Data Set 2
Subsampled Data Set 3
Subsampled Data Set 4
Bagging: Subsample

Original Data

Subsampled Data Set 1
Subsampled Data Set 2
Subsampled Data Set 3
Subsampled Data Set 4
Bagging: Train Components

Original Data

Subsampled Data Set 1

Subsampled Data Set 2

Subsampled Data Set 3

Subsampled Data Set 4

\[ h_1(x) \]

\[ h_2(x) \]

\[ h_3(x) \]

\[ h_4(x) \]
Bagging

• Bootstrap Aggregation
• Draw n new datasets by sampling n << N instances from training data with replacement
• Average over spurious patterns and outliers to reduce variance
• Intuitive, but not theoretically grounded
Boosting

Original Data

Round 1

1  2  3  4  5  6  7  8  9  10
Boosting

Original Data

Round 1

\[ h_1(x) \]
Boosting

Original Data

Round 1

Round 2

$h_1(x)$
Boosting

Original Data

Round 1

\[ h_1(x) \]

Round 2

\[ h_2(x) \]
Boosting

Original Data

Round 1

Round 2

Round 3

$h_1(x)$

$h_2(x)$

$h_3(x)$
Boosting

Original Data

Round 1

1 2 3 4 5 6 7 8 9 10

$\mathbf{x}$

$h_1(\mathbf{x})$

Round 2

1 2 3 4 5 6 7 8 9 10

$\mathbf{x}$

$h_2(\mathbf{x})$

Round 3

1 2 3 4 5 6 7 8 9 10

$\mathbf{x}$

$h_3(\mathbf{x})$

Round 4

1 2 3 4 5 6 7 8 9 10

$\mathbf{x}$

$h_4(\mathbf{x})$
Boosting

Original Data

Round 1

1 2 3 4 5 6 7 8 9 10

$\rightarrow h_1(x)$

Round 2

1 2 3 4 5 6 7 8 9 10

$\rightarrow h_2(x)$

Round 3

1 2 3 4 5 6 7 8 9 10

$\rightarrow h_3(x)$

Round 4

1 2 3 4 5 6 7 8 9 10

$\rightarrow h_4(x)$

Round 5

1 2 3 4 5 6 7 8 9 10

$\rightarrow h_5(x)$
Boosting

• Rather than resampling the data, **weights on some examples** during learning

• Learn a **sequence of simple hypotheses**, each is called a “**weak learner**”

• **Up-weight data** that are difficult
  – Incorrectly classified in the previous round

• **Down-weight data** that are easy
  – Correctly classified in the previous round

• For test data, weight each hypothesis (i.e., classifier) according to its accuracy during training
Figure 4: This figure visualizes the idea of “correct residuals”. We split the train set into $K$ disjoint sets $T_1$ to $T_K$ of equal size and train $K$ different CF models $M_1$ to $M_K$. The first model $M_1$ uses the ratings of the sets $T_2$ to $T_K$ for training and generates predictions for the set $T_1$. The second model $M_2$ excludes the set $T_2$ in the training phase, and calculates predictions for this set. Each rating in the training set is predicted by 1 model. Each rating in the probe and qualifying set is predicted by 34 models. The predictions for the probe and the qualifying set are linear blends of all $K$ models.
First Boosting Algorithms

- **[Schapire’89]**
  - First probable boosting algorithm
- **[Freund’90]**
  - “optimal” algorithm that boosts by majority”
- **[Drucker, Schapire & Simard ‘92]**
  - First experiments using boosting
  - Limited by practical drawbacks
- **[Freund&Schapire’95]**
  - Introduced “AdaBoost” algorithm
  - Strong practical advantages over previous boosting algorithms
- **[Freiedman’99]**
  - Introduced “Gradient Boosting”
Boosting

<table>
<thead>
<tr>
<th>Name</th>
<th>Loss</th>
<th>Derivative</th>
<th>$f^*$</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Squared error</td>
<td>$\frac{1}{2}(y_i - f(x_i))^2$</td>
<td>$y_i - f(x_i)$</td>
<td>$E[y</td>
<td>x_i]$</td>
</tr>
<tr>
<td>Absolute error</td>
<td>$</td>
<td>y_i - f(x_i)</td>
<td>$</td>
<td>sgn($y_i - f(x_i)$)</td>
</tr>
<tr>
<td>Exponential loss</td>
<td>$\exp(-\tilde{y}_i f(x_i))$</td>
<td>$-\tilde{y}_i \exp(-\tilde{y}_i f(x_i))$</td>
<td>$\frac{1}{2} \log \frac{\pi_i}{1-\pi_i}$</td>
<td>AdaBoost</td>
</tr>
<tr>
<td>Logloss</td>
<td>$\log(1 + e^{-\tilde{y}_i f_i})$</td>
<td>$y_i - \pi_i$</td>
<td>$\frac{1}{2} \log \frac{\pi_i}{1-\pi_i}$</td>
<td>LogitBoost</td>
</tr>
</tbody>
</table>

Some commonly used loss functions, their gradients, population minimizers $f^*$, and some algorithms to minimize the loss (Table 16.1, Murphy)

L2Boosting is a special case of Gradient boosting
A Formal Description of Boosting

• given training set \((x_1, y_1), \ldots, (x_m, y_m)\)
• \(y_i \in \{-1, +1\}\) correct label of instance \(x_i \in X\)
• for \(t = 1, \ldots, T\):
  • construct distribution \(D_t\) on \(\{1, \ldots, m\}\)
  • find weak classifier ("rule of thumb") \(h_t : X \rightarrow \{-1, +1\}\)
  
  \[\epsilon_t = \text{Pr}_{i \sim D_t}[h_t(x_i) \neq y_i]\]

• output final/combined classifier \(H_{\text{final}}\)
AdaBoost

- constructing $D_t$:
  - $D_1(i) = 1/m$
  - given $D_t$ and $h_t$:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \ y_i \ h_t(x_i))$$

$\varepsilon_t$: error rate of the classifier in t-th iteration

where $Z_t = \text{normalization factor}$

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_t}{\varepsilon_t} \right) > 0$$

- final classifier:

  - $H_{\text{final}}(x) = \text{sign} \left( \sum_t \alpha_t h_t(x) \right)$
Suppose $\epsilon_t = 0.49$ (bad performance in t-th iteration)

$\alpha_t = 0.02$

for a case that is correctly classified

$D_{t+1}(i) = D_t(i)/Z_t \times e^{-0.02} = D_t(i)/Z_t \times 0.98$

for a case that is incorrectly classified

$D_{t+1}(i) = D_t(i)/Z_t \times e^{0.02} = D_t(i)/Z_t \times 1.02$

Suppose $\epsilon_t = 0.01$ (good performance in t-th iteration)

$\alpha_t = 2.30$

for a case that is correctly classified

$D_{t+1}(i) = D_t(i)/Z_t \times e^{-2.30} = D_t(i)/Z_t \times 0.1$

for a case that is incorrectly classified

$D_{t+1}(i) = D_t(i)/Z_t \times e^{0.02} = D_t(i)/Z_t \times 9.97$
Toy Example

$D_1$

weak classifiers = vertical or horizontal half-planes

Example: one-split decision tree
Round 1

\[ h_1 \]

\[ D_2 \]

\[ \varepsilon_1 = 0.30 \]

\[ \alpha_1 = 0.42 \]
Round 2

\[ h_2 \]

\[ \varepsilon_2 = 0.21 \]

\[ \alpha_2 = 0.65 \]
Round 3

$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$
Final Classifier

\[ H_{\text{final}} = \text{sign} \begin{pmatrix} 0.42 & +0.65 & +0.92 \end{pmatrix} \]
Some commonly used loss functions, their gradients, population minimizers $f^*$, and some algorithms to minimize the loss (Table 16.1, Murphy)

L2Boosting is a special case of Gradient boosting
Gradient Boosting

- Learn a regression predictor
- Compute the error residual
- Learn to predict the residual

One split in decision tree

Learn a simple predictor...

Then try to correct its errors

Mean of points in left region

Average of points in right region
Gradient Boosting

- Learn a regression predictor
- Compute the error residual
- Learn to predict the residual

Combining gives a better predictor… Can try to correct its errors also, & repeat
Gradient Boosting

- Learn sequence of predictors
- Sum of predictions is increasingly accurate
- Predictive function is increasingly complex
Gradient boosting

- Make a set of predictions $\hat{y}[i]$.

- The "error" in our predictions is $J(y, \hat{y})$.
  - For MSE: $J(.) = \sum (y[i] - \hat{y}[i])^2$

- We can "adjust" $\hat{y}$ to try to reduce the error.
  - $\hat{y}[i] = \hat{y}[i] + \alpha f[i]$
  - $f[i] \approx \nabla J(y, \hat{y}) = (y[i]-\hat{y}[i])$ for MSE

- Each learner is estimating the gradient of the loss f’n.
- Gradient descent: take sequence of steps to reduce $J$
  - Sum of predictors, weighted by step size $\alpha$. 

XGBoost: Extreme Gradient Boosting

• XGBoost is developed with both deep consideration in terms of systems optimization and principles in machine learning. The goal of this library is to push the extreme of the computation limits of machines to provide a scalable, portable and accurate library.

• Remark: As of 2019, very widely used classification method, shown superior performance on real-world datasets

• References:
Overfitting

• Boosting is remarkably resistant to overfitting
• In fact, it can often continue to improve even when the training error has gone to zero

Handwritten Character Recognition (Schapire et al)
Boosting Advantages

• Boosting is fast and simple
• It improves the performance many kinds of learning algorithms
• It is easy to program and requires no tuning
• Works well in practice
• Has good theoretical support
• Very popular in combination with decision trees
Summary

• Ensemble methods
  – Combine multiple classifiers to make “better”
  – Bagging: bootstrap data samples
  – Boosting: iteratively reweighting data samples

• AdaBoost
  – Use results of earlier classifiers to know what to work on
  – Weight “hard” examples so we focus on them more
  – Example: viola-Jones for face detection

• Gradient Boosting
  – Use a simple regression model to start
  – Subsequent models predict the error residual of the previous predictions
  – Overall prediction given by a weighted sum of the collection