WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Русский
Edit links

BrownBoost

From Wikipedia, the free encyclopedia

BrownBoost is a boosting algorithm that may be robust to noisy datasets. BrownBoost is an adaptive version of the boost by majority algorithm. As is true for all boosting algorithms, BrownBoost is used in conjunction with other machine learning methods. BrownBoost was introduced by Yoav Freund in 2001.[1]

Contents [hide]

Motivation [edit]

AdaBoost performs well on a variety of datasets; however, it can be shown that AdaBoost does not perform well on noisy data sets.[2] This is a result of AdaBoost's focus on examples that are repeatedly misclassified. In contrast, BrownBoost effectively "gives up" on examples that are repeatedly misclassified. The core assumption of BrownBoost is that noisy examples will be repeatedly mislabeled by the weak hypotheses and non-noisy examples will be correctly labeled frequently enough to not be "given up on." Thus only noisy examples will be "given up on," whereas non-noisy examples will contribute to the final classifier. In turn, if the final classifier is learned from the non-noisy examples, the generalization error of the final classifier may be much better than if learned from noisy and non-noisy examples.

The user of the algorithm can set the amount of error to be tolerated in the training set. Thus, if the training set is noisy (say 10% of all examples are assumed to be mislabeled), the booster can be told to accept a 10% error rate. Since the noisy examples may be ignored, only the true examples will contribute to the learning process.

Algorithm Description [edit]

BrownBoost uses a non-convex potential loss function, thus it does not fit into the AnyBoost framework. The non-convex optimization provides a method to avoid overfitting noisy data sets. However, in contrast to boosting algorithms that analytically minimize a convex loss function (e.g. AdaBoost and LogitBoost), BrownBoost solves a system of two equations and two unknowns using standard numerical methods.

The only parameter of BrownBoost (c in the algorithm) is the "time" the algorithm runs. The theory of BrownBoost states that each hypothesis takes a variable amount of time (t in the algorithm) which is directly related to the weight given to the hypothesis α. The time parameter in BrownBoost is analogous to the number of iterations T in AdaBoost.

A larger value of c means that BrownBoost will treat the data as if it were less noisy and therefore will give up on fewer examples. Conversely, a smaller value of c means that c BrownBoost will treat the data as more noisy and give up on more examples.

During each iteration of the algorithm, a hypothesis is selected with some advantage over random guessing. The weight of this hypothesis α and the "amount of time passed" t during the iteration are simultaneously solved in a system of two non-linear equations (1. uncorrelate hypothesis w.r.t example weights and 2. hold the potential constant) with two unknowns (weight of hypothesis α and time passed t). This can be solved by bisection (as implemented in the JBoost software package) or Newton's method (as described in the original paper by Freund). Once these equations are solved, the margins of each example ($r_i(x_j)$ in the algorithm) and the amount of time remaining s are updated appropriately. This process is repeated until there is no time remaining.

The initial potential is defined to be $\frac{1}{m}\sum_{j=1}^{m} 1-\mathrm{erf}(\sqrt{c}) = 1-\mathrm{erf}(\sqrt{c})$. Since a constraint of each iteration is that the potential be held constant, the final potential is

$\frac{1}{m}\sum_{j=1}^{m} 1-\mathrm{erf}(r_i(x_j)/\sqrt{c}) = 1-\mathrm{erf}(\sqrt{c})$. Thus the final error is *likely* to be near $1-\mathrm{erf}(\sqrt{c})$. However, the final potential function is not the 0-1 loss error function. For the final error to be exactly $1-\mathrm{erf}(\sqrt{c})$, the variance of the loss function must decrease linearly w.r.t. time to form the 0-1 loss function at the end of boosting iterations. This is not yet discussed in the literature and is not in the definition of the algorithm below.

The final classifier is a linear combination of weak hypotheses and is evaluated in the same manner as most other boosting algorithms.

BrownBoost Learning Algorithm Definition [edit]

Input:

- m training examples $(x_1,y_1),\ldots,(x_m,y_m)$ where $x_j \in X,\ y_j \in Y = \{-1,+1\}$
- The parameter c

Initialise:

- $s = c$. The value of s is the amount of time remaining in the game)
- $r_i(x_j) = 0 \quad \forall j$. The value of $r_i(x_j)$ is the margin at iteration i for example x_j.

While $s > 0$:

- Set the weights of each example: $W_i(x_j) = e^{-\frac{(r_i(x_j)+s)^2}{c}}$, where $r_i(x_j)$ is the margin of example x_j
- Find a classifier $h_i : X \to \{-1,+1\}$ such that $\sum_j W_i(x_j)h_i(x_j)y_j > 0$
- Find values α, t that satisfy the equation:
  $\sum_j h_i(x_j)y_j e^{-\frac{(r_i(x_j)+\alpha h_i(x_j)y_j+s-t)^2}{c}} = 0$.
  (Note this is similar to the condition $E_{W_{i+1}}[h_i(x_j)y_j] = 0$ set forth by Schapire and Singer.[3] In this setting, we are numerically finding the $W_{i+1} = \exp(\frac{\ldots}{\ldots})$ such that $E_{W_{i+1}}[h_i(x_j)y_j] = 0$.)
  This update is subject to the constraint
  $\sum\left(\Phi\left(r_i(x_j)+\alpha h(x_j)y_j+s-t\right)-\Phi\left(r_i(x_j)+s\right)\right) = 0$,
  where $\Phi(z) = 1-\mathrm{erf}(z/\sqrt{c})$ is the potential loss for a point with margin $r_i(x_j)$
- Update the margins for each example: $r_{i+1}(x_j) = r_i(x_j)+\alpha h(x_j)y_j$
- Update the time remaining: $s = s - t$

Output: $H(x) = \mathrm{sign}\left(\sum_i \alpha_i h_i(x)\right)$

Empirical Results [edit]

In preliminary experimental results with noisy datasets, BrownBoost outperformed AdaBoost's generalization error; however, LogitBoost performed as well as BrownBoost.[4] An implementation of BrownBoost can be found in the open source software JBoost.

References [edit]

1. ^ Yoav Freund. An adaptive version of the boost by majority algorithm. Machine Learning, 43(3):293–318, June 2001.
2. ^ Dietterich, T. G., (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning, 40 (2) 139-158.
3. ^ Robert Schapire and Yoram Singer. Improved Boosting Using Confidence-rated Predictions. Journal of Machine Learning, Vol 37(3), pages 297-336. 1999
4. ^ Ross A. McDonald, David J. Hand, Idris A. Eckley. An Empirical Comparison of Three Boosting Algorithms on Real Data Sets with Artificial Class Noise. Multiple Classifier Systems, In Series Lecture Notes in Computer Science, pages 35-44, 2003.

See also [edit]

- Boosting
- AdaBoost

- Alternating decision trees
- JBoost

Categories: Classification algorithms | Ensemble learning