Lecture 21: Ensemble classifiers — Boosting

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Data Mining and Machine Learning
August–December 2020

Limitations of classification models

Recall

- Bias : Expressiveness of model limits classification
- Variance: Variation in model based on sample of training data

Overcoming limitations

- Bagging is an effective way to overcome high variance
  - Ensemble models
    - Sequence of models based on independent bootstrap samples
    - Use voting to get an overall classifier
- How can we cope with high bias?

Dealing with bias

- A biased model always makes mistakes
  - Build an ensemble of models to average out mistakes
- Mistakes should be compensated across models in the ensemble
  - How to build a sequence of models, each biased a different way?
  - Again, we assume we have only one set of training data

Boosting

- Build a sequence of weak classifiers $M_1, M_2, \ldots, M_n$ on inputs $D_1, D_2, \ldots, D_n$
  - A weak classifier is any classifier that has error rate strictly below 50%
- Each D_i is a weighted variant of original training data D
  - Initially all weights equal, D_1
  - Going from D_i to D_{i+1} : increase weights where M_i makes mistakes on D_i
  - M_{i+1} will compensate for errors of M_i
- Also, each model M_i gets a weight α_i based on its accuracy on D_i
- Ensemble output
  - Individual classification outcomes are $\{-1, +1\}$
  - Unknown input x: ensemble outcome is weighted sum $\sum_{i=1}^{n} \alpha_i M_i(x)$
  - Check if weighted sum is negative/positive

The boosting algorithm — Adaboost

- Initially, all data items have equal weight

AdaBoost(D, Y, BaseLeaner, k)

$\boxed{\text{Initialize } D_1(w_i) \leftarrow 1/n \text{ for all } i;}$

for $t = 1$ to k **do**

  $f_t \leftarrow$ BaseLearner(D_t);

  $$e_t \leftarrow \sum_{i: f_t(D_t(\mathbf{x}_i)) \neq y_i} D_t(w_i);$$

  if $e_t > ½$ **then**

    $k \leftarrow k - 1;$

    exit-loop

  else

    $\beta_t \leftarrow e_t / (1 - e_t);$

    $$D_{t+1}(w_i) \leftarrow D_t(w_i) \times \begin{cases} \beta_t & \text{if } f_t(D_t(\mathbf{x}_i)) = y_i \\ 1 & \text{otherwise} \end{cases};$$

    $$D_{t+1}(w_i) \leftarrow \frac{D_{t+1}(w_i)}{\sum_{i=1}^{n} D_{t+1}(w_i)}$$

The boosting algorithm — Adaboost

- Initially, all data items have equal weight
- Build a new model and compute its weighted error

AdaBoost(D, Y, BaseLeaner, k)

Initialize $D_1(w_i) \leftarrow 1/n$ for all i;

for $t = 1$ to k **do**

$f_t \leftarrow \text{BaseLearner}(D_t)$;

$e_t \leftarrow \sum_{i: f_t(D_t(\mathbf{x}_i)) \neq y_i} D_t(w_i)$;

if $e_t > ½$ **then**

$k \leftarrow k - 1$;

exit-loop

else

$\beta_t \leftarrow e_t / (1 - e_t)$;

$$D_{t+1}(w_i) \leftarrow D_t(w_i) \times \begin{cases} \beta_t & \text{if } f_t(D_t(\mathbf{x}_i)) = y_i \\ 1 & \text{otherwise} \end{cases};$$

$$D_{t+1}(w_i) \leftarrow \frac{D_{t+1}(w_i)}{\sum_{i=1}^{n} D_{t+1}(w_i)}$$

The boosting algorithm — Adaboost

- Initially, all data items have equal weight
- Build a new model and compute its weighted error
- Discard if error rate is above 50%

AdaBoost(D, Y, BaseLeaner, k)

Initialize $D_1(w_i) \leftarrow 1/n$ for all i;

for $t = 1$ to k **do**

$f_t \leftarrow$ BaseLearner(D_t);

$$e_t \leftarrow \sum_{i: f_t(D_t(\mathbf{x}_i)) \neq y_i} D_t(w_i);$$

if $e_t > \frac{1}{2}$ **then**

$k \leftarrow k - 1$;

exit-loop

else

$\beta_t \leftarrow e_t / (1 - e_t)$;

$$D_{t+1}(w_i) \leftarrow D_t(w_i) \times \begin{cases} \beta_t & \text{if } f_t(D_t(\mathbf{x}_i)) = y_i \\ 1 & \text{otherwise} \end{cases};$$

$$D_{t+1}(w_i) \leftarrow \frac{D_{t+1}(w_i)}{\sum_{i=1}^{n} D_{t+1}(w_i)}$$

The boosting algorithm — Adaboost

- Initially, all data items have equal weight
- Build a new model and compute its weighted error
- Discard if error rate is above 50%
- Damping factor — reduce weight of correct inputs

```
AdaBoost(D, Y, BaseLeaner, k)
    Initialize D_1(w_i) ← 1/n for all i;
    for t = 1 to k do
        f_t ← BaseLearner(D_t);
        e_t ← Σ_{i: f_t(D_t(x_i)) ≠ y_i} D_t(w_i);
        if e_t > ½ then
            k ← k − 1;
            exit-loop
        else
            β_t ← e_t / (1 − e_t);
            D_{t+1}(w_i) ← D_t(w_i) × { β_t  if f_t(D_t(x_i)) = y_i ;  1  otherwise }
            D_{t+1}(w_i) ← D_{t+1}(w_i) / Σ_{i=1}^{n} D_{t+1}(w_i)
```

$$f_t \leftarrow \text{BaseLearner}(D_t); \quad e_t \leftarrow \sum_{i: f_t(D_t(\mathbf{x}_i)) \neq y_i} D_t(w_i);$$

$$\boxed{\beta_t \leftarrow e_t / (1 - e_t);}$$

$$D_{t+1}(w_i) \leftarrow D_t(w_i) \times \begin{cases} \beta_t & \text{if } f_t(D_t(\mathbf{x}_i)) = y_i \\ 1 & \text{otherwise} \end{cases};$$

$$D_{t+1}(w_i) \leftarrow \frac{D_{t+1}(w_i)}{\sum_{i=1}^{n} D_{t+1}(w_i)}$$

The boosting algorithm — Adaboost

- Initially, all data items have equal weight
- Build a new model and compute its weighted error
- Discard if error rate is above 50%
- Damping factor — reduce weight of correct inputs
- Reweight data items and normalize

AdaBoost(D, Y, BaseLeaner, k)

Initialize $D_1(w_i) \leftarrow 1/n$ for all i;
for $t = 1$ to k **do**
  $f_t \leftarrow$ BaseLearner(D_t);
  $e_t \leftarrow \sum_{i: f_t(D_t(\mathbf{x}_i)) \neq y_i} D_t(w_i)$;
  if $e_t > ½$ **then**
    $k \leftarrow k - 1$;
    exit-loop
  else
    $\beta_t \leftarrow e_t / (1 - e_t)$;

$$D_{t+1}(w_i) \leftarrow D_t(w_i) \times \begin{cases} \beta_t & \text{if } f_t(D_t(\mathbf{x}_i)) = y_i \\ 1 & \text{otherwise} \end{cases};$$

$$D_{t+1}(w_i) \leftarrow \frac{D_{t+1}(w_i)}{\sum_{i=1}^{n} D_{t+1}(w_i)}$$

The boosting algorithm — Adaboost

- Initially, all data items have equal weight
- Build a new model and compute its weighted error
- Discard if error rate is above 50%
- Damping factor — reduce weight of correct inputs
- Reweight data items and normalize
- Verdict: weighted sum of individual scores — weights derived from error rate

AdaBoost(D, Y, BaseLeaner, k)

Initialize $D_1(w_i) \leftarrow 1/n$ for all i;

for $t = 1$ to k **do**

$f_t \leftarrow$ BaseLearner(D_t);

$$e_t \leftarrow \sum_{i: f_t(D_t(\mathbf{x}_i)) \neq y_i} D_t(w_i);$$

if $e_t > ½$ **then**

$k \leftarrow k - 1$;

exit-loop

else

$\beta_t \leftarrow e_t / (1 - e_t)$;

$$D_{t+1}(w_i) \leftarrow D_t(w_i) \times \begin{cases} \beta_t & \text{if } f_t(D_t(\mathbf{x}_i)) = y_i \\ 1 & \text{otherwise} \end{cases};$$

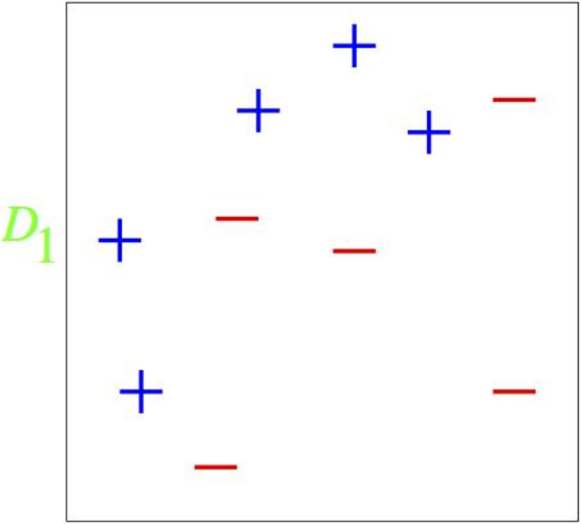
$$D_{t+1}(w_i) \leftarrow \frac{D_{t+1}(w_i)}{\sum_{i=1}^{n} D_{t+1}(w_i)}$$

The boosting algorithm — Adaboost

- Each M_i could be a different type of model
- Can we pick best n out of N weak classifiers?
- Initially all data items have equal weight, select M_1 as model with lowest error rate among N candidates
- Inductively, assume we have selected $M_1, \ldots M_j$, with model weights $\alpha_1, \ldots,$ α_j, and dataset is updated with new weights as D_{j+1}
  - Pick model with lowest error rate on D_{j+1} as M_{j+1}
  - Calculate α_{j+1} based on error rate of M_{j+1}
  - Reweight all training data based on error rate of M_{j+1}
- Note that same model M may be picked in multiple iterations, assigned different weights α

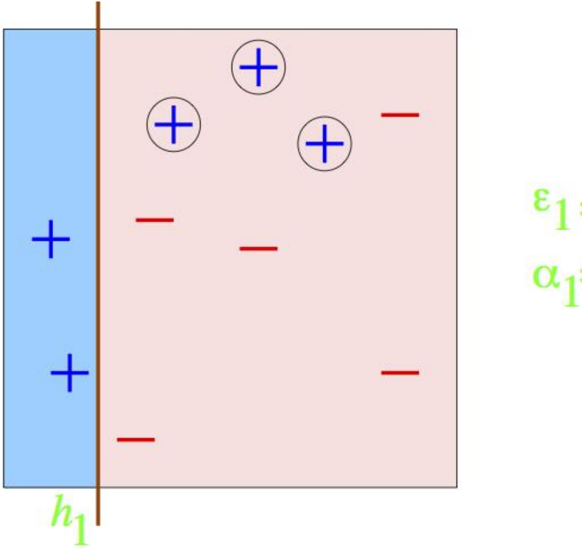
Boosting: An example

- Weak classifiers are horizontal and vertical lines
- Initial training data has equal weights



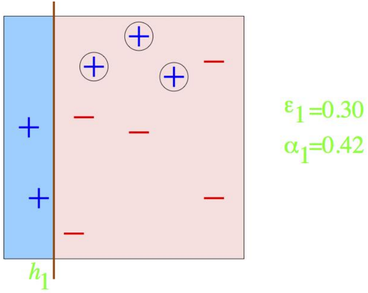
Boosting: An example

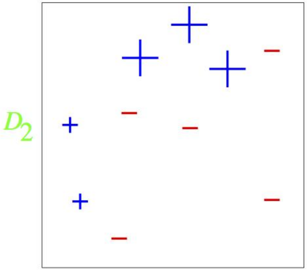
- Weak classifiers are horizontal and vertical lines
- Initial training data has equal weights
- First separator: vertical line



Boosting: An example

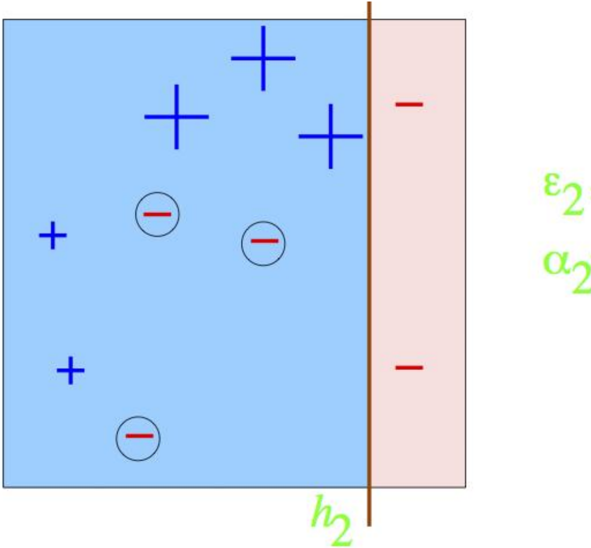
- Weak classifiers are horizontal and vertical lines
- Initial training data has equal weights
- First separator: vertical line
  - Increase weight of misclassified inputs





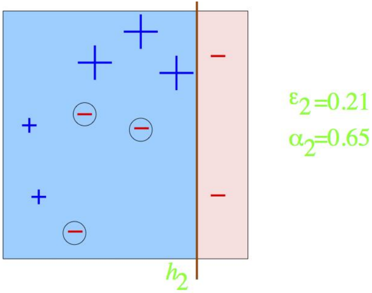
Boosting: An example

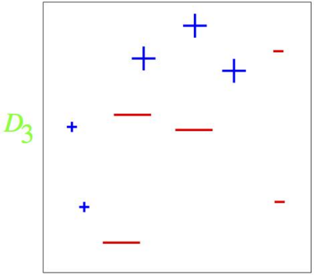
- Weak classifiers are horizontal and vertical lines
- Initial training data has equal weights
- First separator: vertical line
  - Increase weight of misclassified inputs
- Second separator: vertical line



Boosting: An example

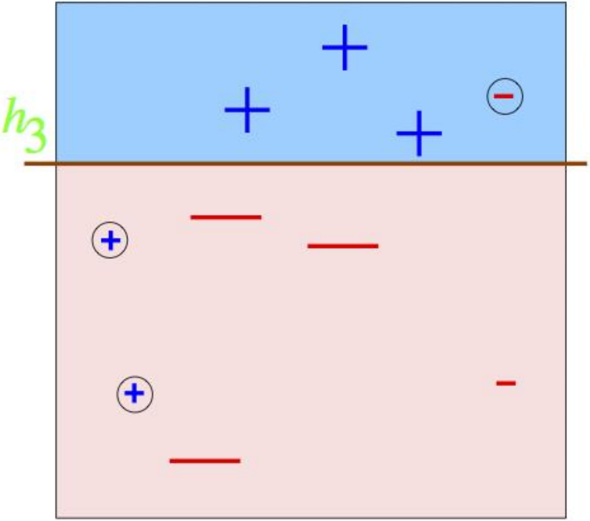
- Weak classifiers are horizontal and vertical lines
- Initial training data has equal weights
- First separator: vertical line
  - Increase weight of misclassified inputs
- Second separator: vertical line
  - Increase weight of misclassified inputs





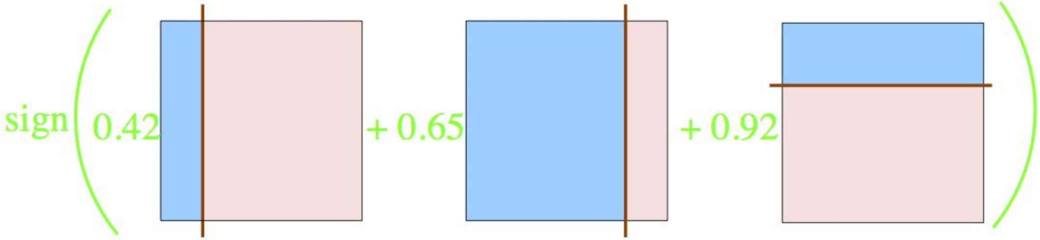
Boosting: An example

- Weak classifiers are horizontal and vertical lines
- Initial training data has equal weights
- First separator: vertical line
  - Increase weight of misclassified inputs
- Second separator: vertical line
  - Increase weight of misclassified inputs
- Third separator: horizontal line

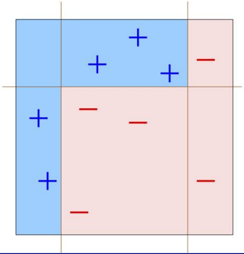


Boosting: An example

- Final classifier is weighted sum of three weak classifiers



- Pictorially



Theoretical analysis — simplified Adaboost

- Each iteration uses the same model building algorithm
- Reweighting factor is uniform at each step, fixed by parameter γ
- Individual models not assigned weights — final classification is a simple majority

Given a sample S of n labeled examples $\mathbf{x}_1, \ldots, \mathbf{x}_n$, initialize each example $\mathbf{x}_i$ to have a weight $w_i = 1$. Let $\mathbf{w} = (w_1, \ldots, w_n)$.

For $t = 1, 2, \ldots, t_0$ do

Call the weak learner on the weighted sample $(S, \mathbf{w})$, receiving hypothesis h_t.

Multiply the weight of each example that was misclassified by h_t by $\alpha = \frac{\frac{1}{2}+\gamma}{\frac{1}{2}-\gamma}$. Leave the other weights as they are.

End

Output the classifier MAJ$(h_1, \ldots, h_{t_0})$ which takes the majority vote of the hypotheses returned by the weak learner. Assume t_0 is odd so there is no tie.

Given a sample S of n labeled examples $\mathbf{x}_1, \ldots, \mathbf{x}_n$, initialize each example $\mathbf{x}_i$ to have a weight $w_i = 1$. Let $\mathbf{w} = (w_1, \ldots, w_n)$.

Theoretical analysis — simplified Adaboost

γ-weak classifier

For any training dataset D, for any assignment of non-negative real weights w_i to $x_i \in D$, classifier correctly labels subset with weight at least $(\frac{1}{2} + \gamma) \sum_{i=1}^{n} w_i$

Theorem

Let A be a γ-weak classification algorithm. Let D be a training dataset of size n. Then $t_0 = O(\frac{1}{\gamma^2} \ln n)$ is sufficient for the boosted classifier $MAJ(h_1, h_2, \ldots, h_{t_0})$ to have training error zero.

For any training dataset D, a γ-weak classifier can be boosted to make correct predictions on all of D.

Theoretical analysis — simplified Adaboost

Proof

- Let m be the number of examples misclassified by final classifier
  - Majority classifier — each such item misclassified at least $t_0/2$ times
  - Each misclassified item has weight at least $\alpha^{t_0/2}$
  - Total weight of misclassified items at least $m\alpha^{t_0/2}$
- Iteration $t+1$, only items misclassifed by h_t are reweighted
  - γ-weak classifier — total weight of misclassified items at most $\frac{1}{2}-\gamma$
- $weight(t)$ – total weight at time t
  - Recall that $\alpha = \frac{\frac{1}{2}+\gamma}{\frac{1}{2}-\gamma}$

$$weight(t+1) \leq \left(\underbrace{\alpha\left(\frac{1}{2}-\gamma\right)}_{misclassified} + \underbrace{\left(\frac{1}{2}+\gamma\right)}_{correct} \right) weight(t) \leq (1+2\gamma) weight(t)$$

Theoretical analysis — simplified Adaboost

Proof

- $weight(0) = n$, so after t_0 iterations, $weight(t_0) \leq n(1+2\gamma)^{t_0}$
- Total weight of misclassified items at least $m\alpha^{t_0/2}$,

  so $m\alpha^{t_0/2} \leq n(1+2\gamma)^{t_0}$
- Rewrite $\alpha = \frac{\frac{1}{2}+\gamma}{\frac{1}{2}-\gamma}$ as $\alpha = \frac{1+2\gamma}{1-2\gamma}$
- $m\alpha^{t_0/2} \leq n(1+2\gamma)^{t_0} \Rightarrow m \leq n(1+2\gamma)^{t_0} \dfrac{(1-2\gamma)^{t_0/2}}{(1+2\gamma)^{t_0/2}}$

  $\Rightarrow m \leq n(1-2\gamma)^{t_0/2}(1+2\gamma)^{t_0/2} \Rightarrow m \leq n(1-4\gamma^2)^{t_0/2} \Rightarrow m \leq n(1-4\gamma^2)^{t_0/2} \Rightarrow$
  $m \leq n(1-4\gamma^2)^{t_0/2}$
- Since $1-x \leq e^{-x}$, $m \leq n\left(e^{-4\gamma^2}\right)^{t_0/2} \Rightarrow m \leq ne^{-2t_0\gamma^2}$
- If $t_0 > \frac{\ln n}{2\gamma^2}$, $m < 1$, so zero training error

Summary

- Boosting provably improves the performance of a biased classifier
- Adjust weights of incorrectly classified inputs to iteratively produce new classifiers that compensate for errors
- Can also do this to select best n of N diverse classifiers
  - Combining expert advice
- Variation: Combining sleeping experts
  - Each expert (classifier) need not classify all inputs
  - For each input, some experts may be "sleeping"