Lecture 11: 14 February, 2023

Pranabendu Misra
Slides by Madhavan Mukund

Data Mining and Machine Learning
January–April 2023

Limitations of classification models

Recall

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

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

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%

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

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

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

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

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

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

- 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**
    $f_t \leftarrow$ 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
        $\beta_t \leftarrow e_t / (1 - e_t)$;
        $D_{t+1}(w_i) \leftarrow D_t(w_i) \times \begin{cases} \beta_t & \text{if } f_t(D_t(\mathbf{x}_i)) = y_i \\ 1 & \text{otherwise} \end{cases}$;
        $D_{t+1}(w_i) \leftarrow \dfrac{D_{t+1}(w_i)}{\sum_{i=1}^{n} D_{t+1}(w_i)}$

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**

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

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

$\quad$ **if** $e_t > ½$ **then**

$\quad\quad k \leftarrow k - 1$;

$\quad\quad$ exit-loop

$\quad$ **else**

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

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

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

- 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)
```

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

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

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

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

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) ← 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)
```

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

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

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
- Final classifier

$$f_{\text{final}}(x) = \arg\max_{y \in Y} \sum_{t : f_t(x) = y} \log \frac{1}{\beta_t}$$

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
    $\beta_t \leftarrow e_t / (1 - e_t)$;
    $D_{t+1}(w_i) \leftarrow D_t(w_i) \times \begin{cases} \beta_t & \text{if } f_t(D_t(\mathbf{x}_i)) = y_i \\ 1 & \text{otherwise} \end{cases}$;
    $D_{t+1}(w_i) \leftarrow \dfrac{D_{t+1}(w_i)}{\sum_{i=1}^{n} D_{t+1}(w_i)}$

The boosting algorithm — Adaboost

- Each M_i could be a different type of model

The boosting algorithm — Adaboost

- Each M_i could be a different type of model
- Can we pick best n out of N weak classifiers?

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

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}

The boosting algorithm — Adaboost

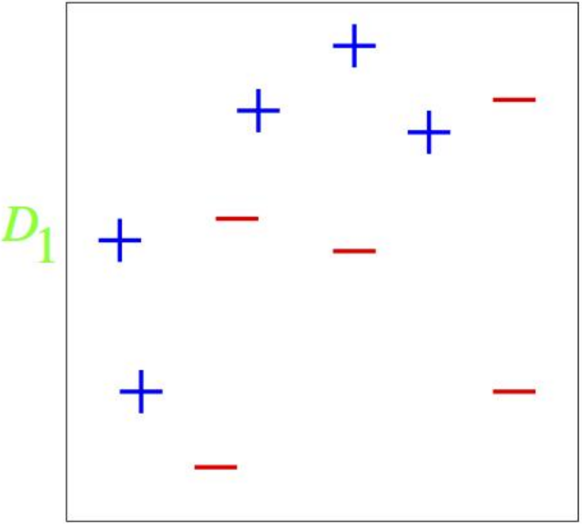
- Each M_i could be a different type of model
- Can we pick best n out of N weak classifiers?
- Initially all data items have equal weight, select M_1 as model with lowest error rate among N candidates
- Inductively, assume we have selected $M_1, \ldots M_j$, with model weights $\alpha_1, \ldots,$ α_j, 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}

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

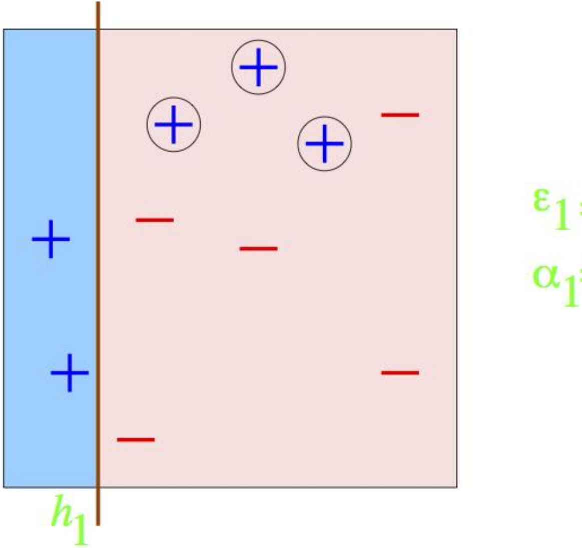
Boosting: An example

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



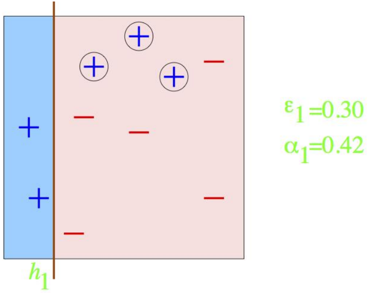
Boosting: An example

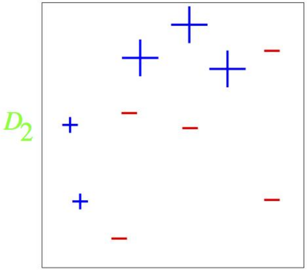
- Weak classifiers are horizontal and vertical lines
- Initial training data has equal weights
- First separator: vertical line



Boosting: An example

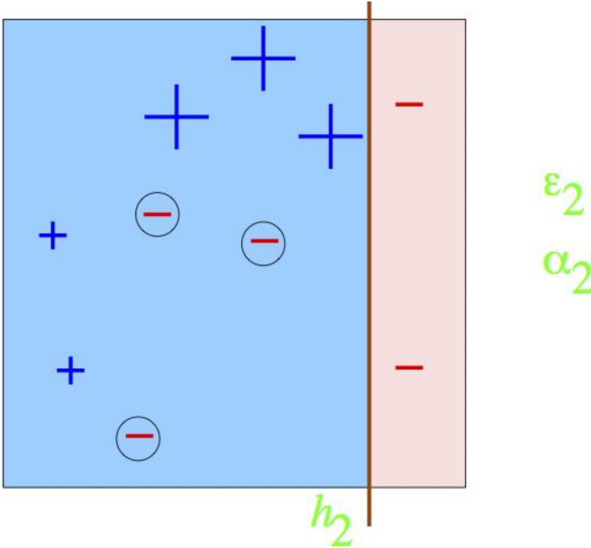
- Weak classifiers are horizontal and vertical lines
- Initial training data has equal weights
- First separator: vertical line
  - Increase weight of misclassified inputs





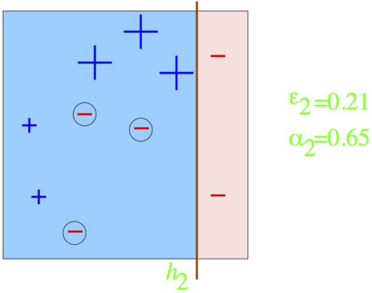
Boosting: An example

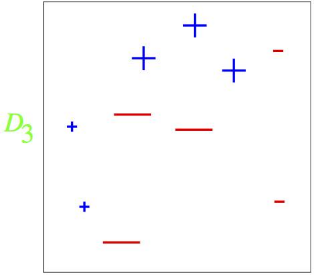
- Weak classifiers are horizontal and vertical lines
- Initial training data has equal weights
- First separator: vertical line
  - Increase weight of misclassified inputs
- Second separator: vertical line



Boosting: An example

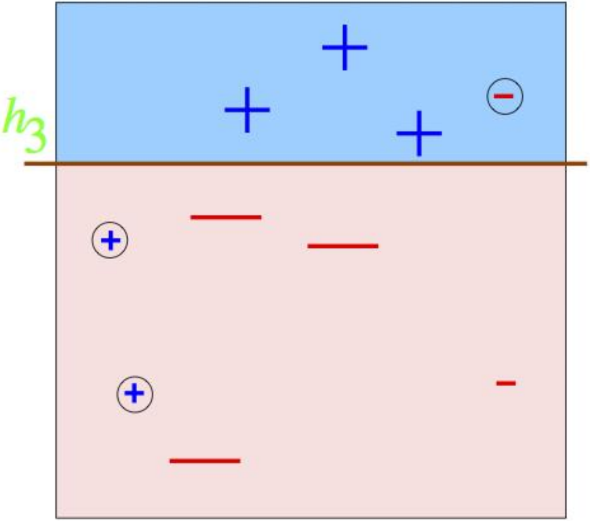
- Weak classifiers are horizontal and vertical lines
- Initial training data has equal weights
- First separator: vertical line
  - Increase weight of misclassified inputs
- Second separator: vertical line
  - Increase weight of misclassified inputs





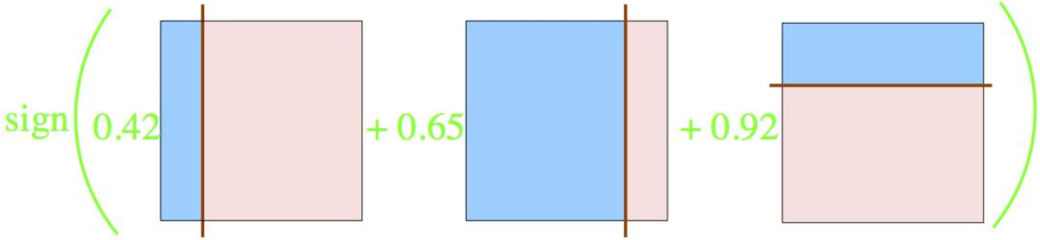
Boosting: An example

- 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



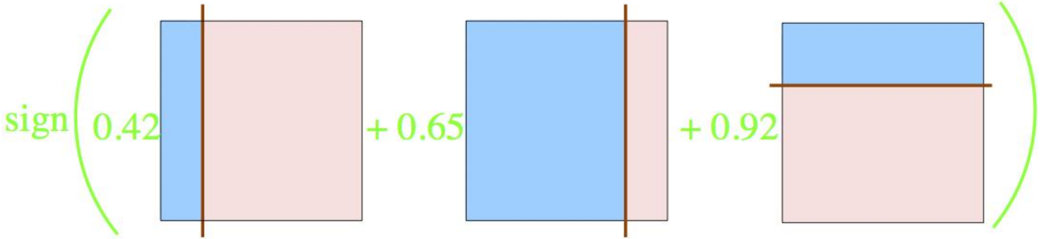
Boosting: An example

- Final classifier is weighted sum of three weak classifiers



Boosting: An example

- Final classifier is weighted sum of three weak classifiers



- Pictorially