Lecture 10 – Ensemble methods

✓ Why do we need ensemble methods?
✓ Bagging
✓ Random forest
✓ Weighted averaging

• **Boosting**
  – Intuitive explanation
  – AdaBoost algorithm
  – Alternative formulations
  – Interpretations
Towards boosting

• Suppose we build M models, each with the same error \( \epsilon \), we then use majority vote

• Is it necessarily the best case if we choose the models to be independent?

• No!
Bounds on Majority Voting tell us more is possible…

These are combinatorial bounds, such that it is not possible to construct a set of instances and predictions that would go out of these bounds.

\[
0 \leq \max \left\{ 0, \frac{M(2\epsilon - 1) + 1}{M + 1} \right\} \leq p(H(x) \neq y) \leq \min \left\{ 1, \frac{2M\epsilon}{M + 1} \right\}
\]

Slide adapted from Gavin Brown
Bounds on Majority Voting tell us more is possible...

\[ 0 \leq \max \left\{ 0, \frac{M(2\epsilon - 1) + 1}{M + 1} \right\} \leq p(H(x) \neq y) \leq \min \left\{ 1, \frac{2M\epsilon}{M + 1} \right\} \]

**Figure 15:** Bounds on majority voting error. The dashed line indicates error in the independent case.

---


---

*Slide adapted from Gavin Brown*
Why do we need ensemble methods?

- Bagging
- Random forest
- Weighted averaging

- Boosting
  - Intuitive explanation
  - AdaBoost algorithm
  - Alternative formulations
  - Interpretations
Sequential construction…

Each model corrects the mistakes of its predecessors.

Slide adapted from Gavin Brown
Define a distribution over the training set, $D_1(i) = \frac{1}{N}$, $\forall i$.

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

- Build a model $h_t$ from the training set, using distribution $D_t$.
- Update $D_{t+1}$ from $D_t$:
  - Increase the weight on examples that $h_t$ incorrectly classifies.
  - Decrease the weight on examples that $h_t$ correctly classifies.

**end for**

For a new testing point $(x', y')$, we take a weighted majority vote from $\{h_1, ... h_T\}$.

---

Slide adapted from Gavin Brown
Slide adapted from Gavin Brown
Half the distribution (~effort) goes on the incorrect examples.

Slide adapted from Gavin Brown
Half the distribution (~effort) goes on the incorrect examples.

Slide adapted from Gavin Brown
Half the distribution (~effort) goes on the incorrect examples.

Slide adapted from Gavin Brown
Requirement

Each model should be slightly better than random guessing on its own dataset, i.e. it is a...

“weak model”
Can your classifier naturally take advantage of a weighted distribution?

- **No**
  - Boosting by "Resampling" (e.g. using linear discriminants)

- **Yes**
  - Boosting by "Reweighting" (e.g. using Naive Bayes)

Slide adapted from Gavin Brown
The Boosting Family of Algorithms

- Very rich history in computational learning theory
- Existence of algorithm was predicted before invention!
- Most well known = “Adaboost”, ADAdptive BOOSTing
- Naturally for two classes, multiclass extensions possible.
- MANY MANY extensions, e.g. regression, ranking etc

Slide adapted from Gavin Brown
Lecture 10 – Ensemble methods

✓ Why do we need ensemble methods?
✓ Bagging
✓ Random forest
✓ Weighted averaging
• Boosting
  ✓ Intuitive explanation
  – AdaBoost algorithm
  – Alternative formulations
  – Interpretations
Adaboost

Define a distribution over the training set, \( D_1(i) = \frac{1}{N}, \forall i \).

for \( t = 1 \) to \( T \) do

Build a classifier \( h_t \) from the training set, using distribution \( D_t \).

Set \( \alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right) \)

Update \( D_{t+1} \) from \( D_t \):

Set \( D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t} \)

end for

\[ H(x') = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x') \right) \]

Weighted majority vote combination

Slide adapted from Gavin Brown
'Adaboost', the most well known boosting algorithm. So, let's get into some details of how Adaboost works.

4.2 More Detail: Adaboost

Adaboost (Adaptive Boosting) is the most well known of the boosting family of procedures. Adaboost is naturally a two-class classifier, and uses a particular formalism: assume each classifier $h_t(x)$ produces an output $\{1, -1\}$, then the decision of a weighted majority voting ensemble can be written as $H(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$. Here, $\alpha_t$ is the weight assigned to voter $t$ in the weighted vote. The detailed algorithm is shown below.

Adaboost: input training data+labels $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, and required number of models $T$

Define a distribution over the training set, $D_1(i) = \frac{1}{N}$, for $t = 1$ to $T$

1. Build a classifier $h_t$ from the training set, using distribution $D_t$.
2. Set $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$
3. Update $D_{t+1}$ from $D_t$:
   - Set $D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$

For a new testing point $(x_0, y_0)$, we take a weighted majority vote, $H(x_0) = \text{sign}\left(\sum_t \alpha_t h_t(x_0)\right)$.

There are two key innovations that Adaboost brings — how $\alpha_t$ and the distribution $D_t$ are set.

The distribution $D_t$ is used by model $h_t$ to learn on the dataset. For models that can naturally take into account different emphases on getting certain points correct, like Naive Bayes, the model can directly use the distribution — this is called the reweighting method. For others, like maybe simple decision stumps, we can re-sample $N$ items (with replacement) from the training set according to the distribution $D_t$.

Notice that at the first iteration when $D_1$ is uniform, this is exactly equivalent to Bagging. However once the first model is built, the distribution is updated to become non-uniform, and Bagging/Boosting diverge. This second method is called resampling.

So how does Adaboost perform? Well this is where it gets interesting — very well indeed. A typical run is below.

![Figure 17: Boosting reduces test error even after training error is zero!](slide)

Slide adapted from Gavin Brown

Game Theory
(Freund & Schapire 1996)

Probabilistic Models
(Lebanon & Lafferty 2001)
(Edakunni, Kovacs & Brown 2011)

Functional Gradient Descent
(Mason et al 2001)

Optimizing Bregman Distances
(Collins 2000)

Dynamical Systems
(Rudin et al 2004)

Probabilistic Models
(Lebanon & Lafferty 2001)
(Edakunni, Kovacs & Brown 2011)

Optimizing Bregman Distances
(Collins 2000)
Notice anything odd?
Train error is ZERO. Test error still decreasing!

More complexity? Not overfitting?


Slide adapted from Gavin Brown
Adaboost in your phone camera….

Slide adapted from Gavin Brown
Typically values for $m$ are $\sqrt{p}$ or even as low as 1. After $B$ such trees \{\(T(x; \Theta_b)\)\} are grown, the random forest (regression) predictor is:

$$\hat{f}_{\text{rf}}(x) = \frac{1}{B} \sum_{b=1}^{B} T(x; \Theta_b).$$

As in Section 10.9 (page 356), $\Theta_b$ characterizes the $b$th random forest tree in terms of split variables, cutpoints at each node, and terminal-node values. Intuitively, reducing $m$ will reduce the correlation between any pair of trees in the ensemble, and hence by (15.1) reduce the variance of the average.

**FIGURE 15.1.** Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each “step” in the figure corresponds to a change in a single misclassification (in a test set of 1536).

Not all estimators can be improved by shaking up the data like this. It seems that highly nonlinear estimators, such as trees, benefit the most. For bootstrapped trees, $\rho$ is typically small ($0.05$ or lower is typical; see Figure 15.9), while $\sigma^2$ is not much larger than the variance for the original tree. On the other hand, bagging does not change linear estimates, such as the sample mean (hence its variance either); the pairwise correlation between bootstrapped means is about 50% (Exercise 15.4).
Netflix challenge - 1 million USD (2006-2009)

- Netflix, an online DVD-rental and online video streaming service
- Task: predict user ratings to films from ratings by other users
- Goal: improve existing method by 10%
- Winner’s solution: ensemble with over 500 heterogeneous models, aggregated with gradient boosted decision trees (we are not covering gradient boosting in the course)

http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/
Main idea behind AdaBoost

• Looking for the best model in the form of a weighted sum of weak models
• Learn the first model
• How to learn the second such that it would correct the errors of the first one?
• Create a weighted dataset where the first model would have 50% accuracy
• Then the second model is forced to do better than the first one
• Repeat the same with 3rd model, 4th, etc
Algorithm AdaBoost($D, T, A$) – train an ensemble of binary classifiers from reweighted training sets.

Input : training data set $D$; ensemble size $T$; learning algorithm $A$.
Output : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$ ; // start with uniform weights

2. for $t = 1$ to $T$ do

3. run $A$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;

4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;

5. if $\epsilon_t \geq 1/2$ then

6. set $T \leftarrow t - 1$ and break

7. end

8. $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$ ; // confidence for this model

9. $w_{(t+1)i} \leftarrow w_{ti} e^{-\alpha_t y_i h_t(x_i)}$ for $i = 1, \ldots, |D|$ ; // update weights

10. $w_{(t+1)i} \leftarrow w_{(t+1)i} / \sum_{j=1}^{|D|} w_{(t+1)j}$ for $i = 1, \ldots, |D|$ ; // renormalize weights

11. end

12. return $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$
Algorithm AdaBoost($D, T, A$) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set $D$; ensemble size $T$; learning algorithm $A$.

**Output** : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$ ;  
   // start with uniform weights

2. **for** $t = 1$ to $T$ **do**

3. run $A$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;

4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;

5. **if** $\epsilon_t \geq 1/2$  
   **then** set $T \leftarrow T/2$  
   **end**

6. $\alpha_t \leftarrow \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$

7. $w_{(t+1)i} \leftarrow w_{ti} e^{-\alpha_t y_i h_t(x_i)}$ for $i = 1, \ldots, |D|$ ;  
   // update weights

8. $w_{(t+1)i} \leftarrow w_{(t+1)i} / \sum_{j=1}^{|D|} w_{(t+1)j}$ for $i = 1, \ldots, |D|$ ;  
   // renormalize weights

9. **end**

10. **return** $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$

**Notation:**

Model $h_t$ outputs binary class +1 or -1 and the true class $y_i$ is also +1 or -1.
**Algorithm** AdaBoost($D, T, \mathcal{A}$) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set $D$; ensemble size $T$; learning algorithm $\mathcal{A}$.

**Output** : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
2. for $t = 1$ to $T$ do
3. run $\mathcal{A}$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
5. if $\epsilon_t \geq 1/2$ then
6. set $T$ to $t$, break;
7. end
8. $\alpha_t = \sqrt{\frac{1}{2} \ln \frac{1}{\epsilon_t}}$;
9. $w_{(t+1)i} = w_{ti} \exp(-\alpha_t I[h_t(x_i) \neq y_i])$;
10. normalize weights $w_{(t+1)i}$;
11. end
12. return $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$

**Notation:** Model $h_t$ outputs binary class $+1$ or $-1$.

On a test instance AdaBoost predicts:

$$\hat{y} = \text{sign}(H(x)) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x) \right)$$
Algorithm AdaBoost($D, T, A$) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set $D$; ensemble size $T$; learning algorithm $A$.

**Output** : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
2. for $t = 1$ to $T$ do
   3. run $A$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
   4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
   5. if $\epsilon_t \geq 1/2$ then
      6. set $T \leftarrow T \times \sqrt{2}$ and break
   end
   7. $\alpha_t \leftarrow \ln(1/\epsilon_t)$; // confidence for this model
   8. $w_{(t+1)i} \leftarrow w_{ti} \cdot \frac{1}{\sqrt{\epsilon_t}}$ for $i = 1, \ldots, |D|$;
   9. end
10. return $\hat{y} = \text{sign}(H(x)) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

**Notation:**
- Model $h_t$ outputs binary class $+1$ or $-1$
- Update weights
- Renormalize weights
- This is weighted majority voting

On a test instance AdaBoost predicts:

$$\hat{y} = \text{sign}(H(x)) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$
Let us consider an example

**Algorithm** AdaBoost\((D, T, \mathcal{A})\) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set \(D\); ensemble size \(T\); learning algorithm \(\mathcal{A}\).

**Output** : weighted ensemble of models.

```
1 \(w_{1i} \leftarrow 1/|D|\) for all \(x_i \in D\); \hspace{1cm} // start with uniform weights
2 \textbf{for} t = 1 \textbf{to} T \textbf{do}
3 \hspace{1cm} \text{run } \mathcal{A} \text{ on } D \text{ with weights } w_{ti} \text{ to produce a model } h_t;
4 \hspace{1cm} \text{calculate weighted error } \epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i];
5 \hspace{1cm} \textbf{if } \epsilon_t \geq 1/2 \textbf{ then}
6 \hspace{2cm} \text{set } T \leftarrow t - 1 \text{ and break}
7 \hspace{1cm} \textbf{end}
8 \hspace{1cm} \alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}; \hspace{1cm} // \text{confidence for this model}
9 \hspace{1cm} w_{(t+1)i} \leftarrow w_{ti} e^{-\alpha_t y_i h_t(x_i)} \text{ for } i = 1, \ldots, |D|; \hspace{1cm} // \text{update weights}
10 \hspace{1cm} w_{(t+1)i} \leftarrow w_{(t+1)i} / \sum_{j=1}^{|D|} w_{(t+1)j} \text{ for } i = 1, \ldots, |D|; \hspace{1cm} // \text{renormalize weights}
11 \textbf{end}
12 \textbf{return } H(x) = \sum_{t=1}^T \alpha_t h_t(x)
```
Let us consider an example

**Algorithm** AdaBoost\((D, T, A)\) – train an ensemble of binary classifiers from reweighted training sets.

**Input**  | training data set \(D\)  | ensemble size \(T\)  | learning algorithm \(A\).
**Output**  | weighted ensemble of models.

1. \(w_{1i} \leftarrow 1/|D|\) for all \(x_i \in D\);  
   // start with uniform weights
2. for \(t = 1\) to \(T\) do
3.     run \(A\) on \(D\) with weights \(w_{ti}\) to produce a model \(h_t\);
4.     calculate weighted error \(\varepsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]\);
5.     if \(\varepsilon_t \geq 1/2\) then
6.         set \(T \leftarrow t - 1\) and break
7.     end
8.     \(\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}\);  
   // confidence for this model
9.     \(w_{(t+1)i} \leftarrow w_{ti} e^{-\alpha_t y_i h_t(x_i)}\) for \(i = 1, \ldots, |D|\);  
   // update weights
10. \(w_{(t+1)i} \leftarrow w_{(t+1)i} / \sum_{j=1}^{|D|} w_{(t+1)j}\) for \(i = 1, \ldots, |D|\);  
    // renormalize weights
11. end
12. return \(H(x) = \sum_{t=1}^T \alpha_t h_t(x)\)

For example:

...
Let us consider an example

Algorithm AdaBoost($D, T, A$) – train an ensemble of binary classifiers from reweighted training sets.

Input : training data set $D$; ensemble size $T$; learning algorithm $A$.

Output : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
2. for $t = 1$ to $T$
3.     run $A$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
4.     calculate weighted error $\varepsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
5.     if $\varepsilon_t \geq 1/2$ then
6.         set $T \leftarrow t - 1$ and break
7.     end
8.     $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$; // confidence for this model
9.     $w_{(t+1)i} \leftarrow w_{ti} e^{-\alpha_t y_i h_t(x_i)}$ for $i = 1, \ldots, |D|$; // update weights
10. $w_{(t+1)i} \leftarrow w_{(t+1)i} / \sum_{j=1}^{|D|} w_{(t+1)j}$ for $i = 1, \ldots, |D|$; // renormalize weights
11. end
12. return $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$

For example:

$T = 3$
Algorithm AdaBoost(D, T, A) – train an ensemble of binary classifiers from reweighted training sets.

Input : training data set D; ensemble size T; learning algorithm A

Output : weighted ensemble of models.

1. \( w_{1i} \leftarrow 1/|D| \) for all \( x_i \in D \); // start with uniform weights

2. for \( t = 1 \) to \( T \) do

3. run A on D with weights \( w_{ti} \) to produce a model \( h_t \);

4. calculate weighted error \( \epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i] \);

5. if \( \epsilon_t \geq 1/2 \) then

6. set \( T \leftarrow t - 1 \) and break

7. end

8. \( \alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t} \); // confidence for this model

9. \( w_{(t+1)i} \leftarrow w_{ti} e^{-\alpha_t y_i h_t(x_i)} \) for \( i = 1, \ldots, |D| \); // update weights

10. \( w_{(t+1)i} \leftarrow w_{(t+1)i} / \sum_{j=1}^{|D|} w_{(t+1)j} \) for \( i = 1, \ldots, |D| \); // renormalize weights

11. end

12. return \( H(x) = \sum_{t=1}^T \alpha_t h_t(x) \)

For example:

Learning algorithm: Decision stump (learns trees with a single decision node)
Let us consider an example

**Algorithm** AdaBoost($D, T, \mathcal{A}$) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set $D$; ensemble size $T$; learning algorithm $\mathcal{A}$.

**Output** : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
2. for $t = 1$ to $T$ do
   3. run $\mathcal{A}$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
   4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
   5. if $\epsilon_t \geq 1/2$ then
      6. set $T \leftarrow t - 1$ and break
   end
   7. $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model
   8. $w_{(t+1)i} \leftarrow w_{ti} e^{-\alpha_t y_i h_t(x_i)}$ for $i = 1, \ldots, |D|$; // update weights
   9. $w_{(t+1)i} \leftarrow w_{(t+1)i} / \sum_{j=1}^{|D|} w_{(t+1)j}$ for $i = 1, \ldots, |D|$; // renormalize weights
10. end
11. return $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$

For example:

$h_1$
Let us consider an example

**Algorithm** AdaBoost\((D, T, \mathcal{A})\) – train an ensemble of binary classifiers from reweighted training sets.

**Input**: training data set \(D\); ensemble size \(T\); learning algorithm \(\mathcal{A}\).

**Output**: weighted ensemble of models.

1. \(w_{1i} \leftarrow 1/|D|\) for all \(x_i \in D\); // start with uniform weights
2. for \(t = 1\) to \(T\) do
3. run \(\mathcal{A}\) on \(D\) with weights \(w_{ti}\) to produce a model \(h_t\);
4. calculate weighted error \(\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]\);
5. if \(\epsilon_t \geq 1/2\) then
6. set \(T \leftarrow t - 1\) and break
7. end
8. \(\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}\); // confidence for this model
9. \(w_{(t+1)i} \leftarrow w_{ti} e^{-\alpha_t y_i h_t(x_i)}\) for \(i = 1, \ldots, |D|\); // update weights
10. \(w_{(t+1)i} \leftarrow w_{(t+1)i} / \sum_{j=1}^{|D|} w_{(t+1)j}\) for \(i = 1, \ldots, |D|\); // renormalize weights
11. end
12. return \(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\)

For example:
Let us consider an example

**Algorithm** AdaBoost\((D, T, \mathcal{A})\) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set \(D\); ensemble size \(T\); learning algorithm \(\mathcal{A}\).

**Output** : weighted ensemble of models.

1. \(w_{1i} \leftarrow 1/|D|\) for all \(x_i \in D\); \hspace{1cm} // start with uniform weights
2. \textbf{for } \(t = 1\) to \(T\) \textbf{do}
3. \hspace{1cm} run \(\mathcal{A}\) on \(D\) with weights \(w_{ti}\) to produce a model \(h_t\);
4. \hspace{1cm} calculate weighted error \(\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]\);
5. \hspace{1cm} \textbf{if } \epsilon_t \geq 1/2 \textbf{ then}
6. \hspace{2cm} set \(T \leftarrow t - 1\) and break
7. \hspace{1cm} \textbf{end}
8. \hspace{1cm} \(\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}\); \hspace{1cm} // confidence for this model
9. \hspace{1cm} \(w_{(t+1)i} \leftarrow w_{ti} e^{-\alpha_t y_i h_t(x_i)}\) for \(i = 1, \ldots, |D|\); \hspace{1cm} // update weights
10. \hspace{1cm} \(w_{(t+1)i} \leftarrow w_{(t+1)i} / \sum_{j=1}^{|D|} w_{(t+1)j}\) for \(i = 1, \ldots, |D|\); \hspace{1cm} // renormalize weights
11. \textbf{end}
12. \textbf{return } H(x) = \sum_{t=1}^T \alpha_t h_t(x)

For example:
Let us consider an example

**Algorithm** AdaBoost\((D, T, \mathcal{A})\) – train an ensemble of binary classifiers from reweighted training sets.

**Input**: training data set \(D\); ensemble size \(T\); learning algorithm \(\mathcal{A}\).

**Output**: weighted ensemble of models.

1. \(w_{1i} \leftarrow 1/|D|\) for all \(x_i \in D\); // start with uniform weights
2. for \(t = 1\) to \(T\) do
3. run \(\mathcal{A}\) on \(D\) with weights \(w_{ti}\) to produce a model \(h_t\);
4. calculate weighted error \(\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i];\)
5. if \(\epsilon_t \geq 1/2\) then
6. set \(T \leftarrow t - 1\) and break
7. end
8. \(\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t};\) // confidence for this model
9. \(w_{(t+1)i} \leftarrow w_{ti} e^{-\alpha_t y_i h_t(x_i)}\) for \(i = 1, \ldots, |D|;\) // update weights
10. \(w_{(t+1)i} \leftarrow w_{(t+1)i} / \sum_{j=1}^{|D|} w_{(t+1)j}\) for \(i = 1, \ldots, |D|;\) // renormalize weights
11. end
12. return \(H(x) = \sum_{t=1}^T \alpha_t h_t(x)\)

\[
H(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)
\]
Lecture 10 – Ensemble methods

✓ Why do we need ensemble methods?
✓ Bagging
✓ Random forest
✓ Weighted averaging

• Boosting
  ✓ Intuitive explanation
  ✓ AdaBoost algorithm
  – Alternative formulations
  – Interpretations
**Algorithm** AdaBoost($D, T, A$) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set $D$; ensemble size $T$; learning algorithm $A$.
**Output** : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
2. for $t = 1$ to $T$ do
   3. run $A$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
   4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
   5. if $\epsilon_t \geq 1/2$ then
      6. set $T' = t$ and break
   end
   7. $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model
   8. $w_{(t+1)i} \leftarrow w_{ti} e^{-\alpha_t y_i h_t(x_i)}$ for $i = 1, \ldots, |D|$; // update weights
   9. $w_{(t+1)i} \leftarrow w_{(t+1)i} / \sum_{j=1}^{|D|} w_{(t+1)j}$ for $i = 1, \ldots, |D|$; // renormalize weights
3. end
4. return $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$

Let us consider many alternative ways of writing down the same AdaBoost algorithm.
Algorithm AdaBoost\((D, T, \mathcal{A})\) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set \(D\); ensemble size \(T\); learning algorithm \(\mathcal{A}\).

**Output** : weighted ensemble of models.

\[
\begin{align*}
1 & \quad w_{1i} \leftarrow 1/|D| \text{ for all } x_i \in D ; \quad \text{// start with uniform weights} \\
2 & \quad \text{for } t = 1 \text{ to } T \text{ do} \\
3 & \quad \quad \text{run } \mathcal{A} \text{ on } D \text{ with weights } w_{ti} \text{ to produce a model } h_t; \\
4 & \quad \quad \text{calculate weighted error } \epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]; \\
5 & \quad \quad \text{if } \epsilon_t \geq 1/2 \text{ then} \\
6 & \quad \quad \quad \quad \text{set } T \sqrt{\frac{1}{2\ln \frac{1-\epsilon_t}{\epsilon_t}}} \text{ and break} \\
7 & \quad \quad \end{align*}
\]

\[\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}; \quad \text{// confidence for this model} \]

\[w_{(t+1)i} \leftarrow w_{ti} e^{-\alpha_t y_i h_t(x_i)} \text{ for } i = 1, \ldots, |D| ; \quad \text{// update weights} \]

\[w_{(t+1)i} \leftarrow w_{(t+1)i} / \sum_{j=1}^{|D|} w_{(t+1)j} \text{ for } i = 1, \ldots, |D| ; \quad \text{// renormalize weights} \]

\[\text{end} \]

\[\text{return } H(x) = \sum_{t=1}^T \alpha_t h_t(x) \]

Let us consider many alternative ways of writing down the same AdaBoost algorithm.
Algorithm AdaBoost($D, T, \mathcal{A}$) – train an ensemble of binary classifiers from reweighted training sets.

**Input**: training data set $D$; ensemble size $T$; learning algorithm $\mathcal{A}$.

**Output**: weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$;  // start with uniform weights
2. **for** $t = 1$ to $T$ **do**
3. 
   run $\mathcal{A}$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
4. 
   calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
5. 
   **if** $\epsilon_t \geq 1/2$ **then**
   6. 
      set $T \leftarrow t - 1$ and break
   **end**
    
7. 
   $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$;  // confidence for this model
8. 
   $w_{(t+1)i} \leftarrow \frac{w_{ti} e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$ for $i = 1, \ldots, |D|$;  // update and renormalize weights
9. **end**
10. **return** $H(x) = \\sum_{t=1}^T \alpha_t h_t(x)$

Here $Z_t$ is the normalizing constant such that the new weights add up to 1.
**Algorithm** AdaBoost($D, T, \mathcal{A}$) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set $D$; ensemble size $T$; learning algorithm $\mathcal{A}$.  
**Output** : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
2. for $t = 1$ to $T$ do
3. run $\mathcal{A}$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
5. if $\epsilon_t \geq 1/2$ then
6. Equal to +1 if correctly classified and -1 if incorrectly classified
7. end
8. $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1}{\epsilon_t}$; // confidence for this model
9. $w_{(t+1)i} \leftarrow \frac{w_{ti} e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$ for $i = 1, \ldots, |D|$; // update and renormalize weights
10. end
11. return $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$
**Algorithm** AdaBoost($D, T, \mathcal{A}$) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set $D$; ensemble size $T$; learning algorithm $\mathcal{A}$.

**Output** : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
2. for $t = 1$ to $T$ do
3. run $\mathcal{A}$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
5. if $\epsilon_t \geq 1/2$ then
6. set $T \leftarrow t - 1$ and break
7. end
8. $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model
9. $w_{(t+1)i} \leftarrow \frac{w_{ti} e^{\alpha_t}}{Z_t}$ for misclassified instances $x_i \in D$; // increase
10. $w_{(t+1)j} \leftarrow \frac{w_{tj} e^{-\alpha_t}}{Z_t}$ for correctly classified instances $x_j \in D$; // decrease
11. end
12. return $H(x) = \sum_{i=1}^{T} \alpha_t h_t(x)$
**Algorithm** AdaBoost($D, T, \mathcal{A}$) — train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set $D$; ensemble size $T$; learning algorithm $\mathcal{A}$.

**Output** : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
2. **for** $t = 1$ to $T$ **do**
   3. run $\mathcal{A}$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
   4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
   5. **if** $\epsilon_t \geq 1/2$ **then**
      6. set $T \leftarrow t - 1$ and break
   7. **end**
   8. $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model
   9. $w_{(t+1)i} \leftarrow \frac{w_{ti} e^{\alpha_t}}{Z_t}$ for misclassified instances $x_i \in D$; // increase
10. $w_{(t+1)j} \leftarrow \frac{w_{tj} e^{-\alpha_t}}{Z_t}$ for correctly classified instances $x_j \in D$; // decrease
11. **end**
12. **return** $H(x) = \sum_{i=1}^T \alpha_t h_t(x)$
**Algorithm** AdaBoost\((D, T, \mathcal{A})\) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set \(D\); ensemble size \(T\); learning algorithm \(\mathcal{A}\).

**Output** : weighted ensemble of models.

1. \(w_{1i} \leftarrow 1/|D|\) for all \(x_i \in D\); // start with uniform weights
2. **for** \(t = 1\) to \(T\) **do**
   3. run \(\mathcal{A}\) on \(D\) with weights \(w_{ti}\) to produce a model \(h_t\);
   4. calculate weighted error \(\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]\);
   5. **if** \(\epsilon_t \geq 1/2\) **then**
      6. set \(T \leftarrow t - 1\) and break
   **end**
   7. \(\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}\); // confidence for this model
   8. \(w_{(t+1)i} \leftarrow \frac{w_{ti} \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t}\) for misclassified instances \(x_i \in D\); // increase
   9. \(w_{(t+1)j} \leftarrow \frac{w_{tj} \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}}{Z_t}\) for correctly classified instances \(x_j \in D\); // decrease
10. **end**
11. **return** \(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\)
Algorithm AdaBoost\((D, T, \mathcal{A})\) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set \(D\); ensemble size \(T\); learning algorithm \(\mathcal{A}\).

**Output** : weighted ensemble of models.

1. \(w_{1i} \leftarrow 1/|D| \) for all \(x_i \in D\); \hspace{1cm} // start with uniform weights
2. **for** \(t = 1\) to \(T\) **do**
3. \hspace{1cm} run \(\mathcal{A}\) on \(D\) with weights \(w_{ti}\) to produce a model \(h_t\);
4. \hspace{1cm} calculate weighted error \(\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]\);
5. \hspace{1cm} **if** \(\epsilon_t \geq 1/2\) **then**
6. \hspace{2cm} set \(T \leftarrow t - 1\) and break
7. **end**
8. \(\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t} \) ; \hspace{1cm} // confidence for this model
9. \hspace{1cm} \(w_{(t+1)i} \leftarrow \frac{w_{ti} \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t} \) for misclassified instances \(x_i \in D\) ; \hspace{1cm} // increase
10. \hspace{1cm} \(w_{(t+1)j} \leftarrow \frac{w_{tj} \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}}{Z_t} \) for correctly classified instances \(x_j \in D\) ; \hspace{1cm} // decrease
11. **end**
12. **return** \(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\)
Algorithm AdaBoost($D, T, A$) – train an ensemble of binary classifiers from reweighted training sets.

**Input**: training data set $D$; ensemble size $T$; learning algorithm $A$.

**Output**: weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; \hspace{1cm} // start with uniform weights
2. **for** $t = 1$ to $T$ **do**
3. \hspace{1cm} run $A$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
4. \hspace{1cm} calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
5. \hspace{1cm} **if** $\epsilon_t \geq 1/2$ **then**
6. \hspace{2cm} set $T \leftarrow t - 1$ and break
7. **end**
8. \hspace{1cm} $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; \hspace{1cm} // confidence for this model
9. \hspace{1cm} $w_{(t+1)i} \leftarrow \frac{w_{ti} \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t}$ for misclassified instances $x_i \in D$; \hspace{1cm} // increase
10. \hspace{1cm} $w_{(t+1)j} \leftarrow \frac{w_{tj} \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}}{Z_t}$ for correctly classified instances $x_j \in D$; \hspace{1cm} // decrease
11. **end**
12. **return** $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$
Algorithm AdaBoost($D, T, A$) – train an ensemble of binary classifiers from reweighted training sets.

**Input**: training data set $D$; ensemble size $T$; learning algorithm $A$.

**Output**: weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
2. **for** $t = 1$ to $T$ **do**
3.  **run** $A$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
4.  calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
5.  **if** $\epsilon_t \geq 1/2$ **then**
6.    set $T \leftarrow t - 1$ and break
7. **end**
8.  $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model
9.  $w_{(t+1)i} \leftarrow \frac{w_{ti} \sqrt{\frac{1-\epsilon_t}{\epsilon_t^2(1-\epsilon_t)}}}{Z_t/\sqrt{\epsilon_t(1-\epsilon_t)}}$ for misclassified instances $x_i \in D$; // increase
10. $w_{(t+1)j} \leftarrow \frac{w_{tj} \sqrt{\frac{\epsilon_t}{(1-\epsilon_t)^2 \epsilon_t}}}{Z_t/\sqrt{\epsilon_t(1-\epsilon_t)}}$ for correctly classified instances $x_j \in D$; // decrease
11. **end**
12. **return** $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$
**Algorithm** AdaBoost($D, T, \mathcal{A}$) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set $D$; ensemble size $T$; learning algorithm $\mathcal{A}$.

**Output** : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights

2. for $t = 1$ to $T$ do

3. run $\mathcal{A}$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;

4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;

5. if $\epsilon_t \geq 1/2$ then

6. set $T \leftarrow t - 1$ and break

7. end

8. $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model

9. $w_{(t+1)i} \leftarrow \frac{w_{ti}/\epsilon_t}{Z_t/\sqrt{\epsilon_t(1-\epsilon_t)}}$ for misclassified instances $x_i \in D$; // increase

10. $w_{(t+1)j} \leftarrow \frac{w_{tj}/(1-\epsilon_t)}{Z_t/\sqrt{\epsilon_t(1-\epsilon_t)}}$ for correctly classified instances $x_j \in D$; // decrease

11. end

12. **return** $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$
Algorithm AdaBoost\((D, T, \mathcal{A})\) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set \(D\); ensemble size \(T\); learning algorithm \(\mathcal{A}\).

**Output** : weighted ensemble of models.

1. \(w_{1i} \leftarrow 1/|D|\) for all \(x_i \in D\); \hspace{1cm} // start with uniform weights
2. **for** \(t = 1\) to \(T\) **do**
   3. run \(\mathcal{A}\) on \(D\) with weights \(w_{ti}\) to produce a model \(h_t\);
   4. calculate weighted error \(\varepsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i];\)
   5. **if** \(\varepsilon_t \geq 1/2\) **then**
      6. set \(T \leftarrow t - 1\) and break
   **end**
   7. \(\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t};\) \hspace{1cm} // confidence for this model
   8. \(w_{(t+1)i} \leftarrow \frac{w_{ti}/\varepsilon_t}{Z_t/\sqrt{\varepsilon_t(1-\varepsilon_t)}}\) for misclassified instances \(x_i \in D\); \hspace{1cm} // increase
   9. \(w_{(t+1)j} \leftarrow \frac{w_{tj}/(1-\varepsilon_t)}{Z_t/\sqrt{\varepsilon_t(1-\varepsilon_t)}}\) for correctly classified instances \(x_j \in D\); \hspace{1cm} // decrease
10. **end**
11. **return** \(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\)
Algorithm AdaBoost($D, T, \mathcal{A}$) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set $D$; ensemble size $T$; learning algorithm $\mathcal{A}$.

**Output** : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; \hfill // start with uniform weights

2. for $t = 1$ to $T$ do

3. run $\mathcal{A}$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;

4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;

5. if $\epsilon_t \geq 1/2$ then

6. \hspace{1em} set $T \leftarrow t - 1$ and break

7. end

8. $\alpha_t \leftarrow \frac{1}{Z_t \sqrt{\epsilon_t(1-\epsilon_t)}}$; \hfill // confidence for this model

9. $w_{(t+1)i} \leftarrow \frac{w_{ti} / \epsilon_t}{Z_t / \sqrt{\epsilon_t(1-\epsilon_t)}}$ for misclassified instances $x_i \in D$; \hfill // increase

10. $w_{(t+1)j} \leftarrow \frac{w_{tj} / (1-\epsilon_t)}{Z_t / \sqrt{\epsilon_t(1-\epsilon_t)}}$ for correctly classified instances $x_j \in D$; \hfill // decrease

11. end

12. return $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$

---

Since we are renormalizing anyways, we can omit the constant term with square root.
**Algorithm** AdaBoost($D, T, \mathcal{A}$) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set $D$; ensemble size $T$; learning algorithm $\mathcal{A}$.

**Output** : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
2. **for** $t = 1$ to $T$ **do**
3. run $\mathcal{A}$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
5. **if** $\epsilon_t \geq 1/2$ **then**
   6. set $T \leftarrow t - 1$ and break
6. **end**
7. $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model
8. $w_{(t+1)i} \leftarrow \frac{w_{ti}/\epsilon_t}{Z_t}$ for misclassified instances $x_i \in D$; // increase
9. $w_{(t+1)j} \leftarrow \frac{w_{tj}/(1-\epsilon_t)}{Z_t}$ for correctly classified instances $x_j \in D$; // decrease
10. **end**
11. **return** $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$
Algorithm AdaBoost($D, T, \mathcal{A}$) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set $D$; ensemble size $T$; learning algorithm $\mathcal{A}$.

**Output** : weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$;  \hspace{1cm}  // start with uniform weights
2. **for** $t = 1$ to $T$ **do**
3. \hspace{0.5cm} run $\mathcal{A}$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
4. \hspace{0.5cm} calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
5. \hspace{0.5cm} **if** $\epsilon_t \geq 1/2$ **then**
6. \hspace{1cm} set $T \leftarrow t - 1$ and break
7. **end**
8. $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$;  \hspace{1cm}  // confidence for this model
9. $w_{(t+1)i} \leftarrow \frac{w_{ti}/\epsilon_t}{Z_t}$ for misclassified instances $x_i \in D$;  \hspace{1cm}  // increase
10. $w_{(t+1)j} \leftarrow \frac{w_{tj}/(1-\epsilon_t)}{Z_t}$ for correctly classified instances $x_j \in D$;  \hspace{1cm}  // decrease
11. **end**
12. **return** $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$
Algorithm AdaBoost\((D, T, \mathcal{A})\) – train an ensemble of binary classifiers from reweighted training sets.

**Input**: training data set \(D\); ensemble size \(T\); learning algorithm \(\mathcal{A}\).

**Output**: weighted ensemble of models.

1. \(w_{1i} \leftarrow 1/|D|\) for all \(x_i \in D\); // start with uniform weights

2. for \(t = 1\) to \(T\) do

3. run \(\mathcal{A}\) on \(D\) with weights \(w_{ti}\) to produce a model \(h_t\);

4. calculate weighted error \(\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]\);

5. if \(\epsilon_t > 1/2\) then

   Using the definition of weighted error it can be shown that \(Z_t = 2\)

   \(\alpha_t = \frac{\epsilon_t}{2 \ln \frac{1}{\epsilon_t}}\); // confidence for this model

6. end

7. \(w_{(t+1)i} \leftarrow \frac{w_{ti} \epsilon_t}{Z_t}\) for misclassified instances \(x_i \in D\); // increase

8. \(w_{(t+1)j} \leftarrow \frac{w_{tj} (1-\epsilon_t)}{Z_t}\) for correctly classified instances \(x_j \in D\); // decrease

9. end

10. return \(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\)
**Algorithm** AdaBoost($D, T, \mathcal{A}$) – train an ensemble of binary classifiers from reweighted training sets.

**Input**: training data set $D$; ensemble size $T$; learning algorithm $\mathcal{A}$.

**Output**: weighted ensemble of models.

1. $w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
2. for $t = 1$ to $T$ do
3. run $\mathcal{A}$ on $D$ with weights $w_{ti}$ to produce a model $h_t$;
4. calculate weighted error $\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]$;
5. if $\epsilon_t \geq 1/2$ then
6. set $T \leftarrow t - 1$ and break
7. end
8. $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model
9. $w_{(t+1)i} \leftarrow \frac{w_{ti}}{2\epsilon_t}$ for misclassified instances $x_i \in D$; // increase weight
10. $w_{(t+1)j} \leftarrow \frac{w_{tj}}{2(1-\epsilon_t)}$ for correctly classified instances $x_j \in D$; // decrease
11. end
12. return $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$
Algorithm AdaBoost\((D, T, \mathcal{A})\) – train an ensemble of binary classifiers from reweighted training sets.

**Input** : training data set \(D\); ensemble size \(T\); learning algorithm \(\mathcal{A}\).

**Output** : weighted ensemble of models.

1. \(w_{1i} \leftarrow 1/|D|\) for all \(x_i \in D\); // start with uniform weights

2. **for** \(t = 1\) to \(T\) **do**
   3. run \(\mathcal{A}\) on \(D\) with weights \(w_{ti}\) to produce a model \(h_t\);
   4. calculate weighted error \(\epsilon_t = \sum_{i=1}^{|D|} w_{ti} I[h_t(x_i) \neq y_i]\);
   5. **if** \(\epsilon_t > 1/2\) **then**
      - As we saw, there are many equivalent ways of writing down the AdaBoost algorithm
      6. \(\alpha_t \leftarrow \frac{1}{2} \ln \frac{\epsilon_t}{1 - \epsilon_t} \); // confidence for this model
      7. \(w_{(t+1)i} \leftarrow \frac{w_{ti}}{2\epsilon_t}\) for misclassified instances \(x_i \in D\); // increase weight
      8. \(w_{(t+1)j} \leftarrow \frac{w_{tj}}{2(1-\epsilon_t)}\) for correctly classified instances \(x_j \in D\); // decrease
   6. **end**
12. **return** \(H(x) = \sum_{t=1}^T \alpha_t h_t(x)\)
If we apply AdaBoost on base learners that have linear decision boundaries then

A. Ensemble has also a linear decision boundary

B. Ensemble might be linear or non-linear

C. Ensemble has a non-linear decision boundary

D. I don’t know
Lecture 10 – Ensemble methods

- Why do we need ensemble methods?
- Bagging
- Random forest
- Weighted averaging
- Boosting
  - Intuitive explanation
  - AdaBoost algorithm
  - Alternative formulations
  - Interpretations
Interpretation of AdaBoost weight updates

- At each iteration of AdaBoost, the weights are updated such that:
  - half of total weight is on misclassified instances
  - the other half on correctly classified instances
- This ensures that the current ensemble would have weighted error 50%
- As weak learner is expected to achieve less than 50% weighted error, it is also expected to learn something new (that the ensemble does not 'know' yet)
AdaBoost and exponential loss

- AdaBoost can be interpreted as greedily minimizing exponential loss at each step

\[
\text{ExpLoss} = \prod_{i=1}^{D} e^{-y_i H(x_i)}
\]
Lecture 10 – Ensemble methods

✓ Why do we need ensemble methods?
✓ Bagging
✓ Random forest
✓ Weighted averaging
✓ Boosting
  ✓ Intuitive explanation
  ✓ AdaBoost algorithm
  ✓ Alternative formulations
  ✓ Interpretations