Boosting: Wisdom of the Crowd

Christian Peters

January 29, 2021

What you will know

What you will know

→ The idea behind boosting

What you will know

$\rightarrow$ The idea behind boosting

$\rightarrow$ How to create a strong and efficient learning algorithm

What you will know

$\rightarrow$ The idea behind boosting

$\rightarrow$ How to create a strong and efficient learning algorithm

$\rightarrow$ What is AdaBoost and why is it so successful?

Let’s talk about training a model

How to train a machine learning model

What we have learned so far...

How to train a machine learning model

What we have learned so far...

- We have to pick a hypothesis class $\mathcal{H}$

What we have learned so far...

- We have to pick a hypothesis class $\mathcal{H}$
- $\mathcal{H}$ can't be too complex (VC dim needs to be finite)

How to train a machine learning model

What we have learned so far...

- We have to pick a hypothesis class $\mathcal{H}$
- $\mathcal{H}$ can't be too complex (VC dim needs to be finite)
- We need enough training data (more than some threshold $m_{\mathcal{H}}$)

How to train a machine learning model

What we have learned so far...

- We have to pick a hypothesis class $\mathcal{H}$
- $\mathcal{H}$ can't be too complex (VC dim needs to be finite)
- We need enough training data (more than some threshold $m_{\mathcal{H}}$)
- Then we use ERM to pick the best $h \in \mathcal{H}$ that minimizes the empirical error

How to train a machine learning model

What we have learned so far...

- We have to pick a hypothesis class $\mathcal{H}$
- $\mathcal{H}$ can't be too complex (VC dim needs to be finite)
- We need enough training data (more than some threshold $m_{\mathcal{H}}$)
- Then we use ERM to pick the best $h \in \mathcal{H}$ that minimizes the empirical error

But there is one problem...

The problem with ERM

$\boxed{\text{ERM can be hard.}}$

$\boxed{\text{ERM can be hard.}}$

- Depending on $\mathcal{H}$, the optimization problem can become arbitrarily complex

$\boxed{\text{ERM can be hard.}}$

- Depending on $\mathcal{H}$, the optimization problem can become arbitrarily complex
- e.g. implementing ERM for halfspaces in the non-separable case is computationally hard (chapter 9)

$\boxed{\text{ERM can be hard.}}$

- Depending on $\mathcal{H}$, the optimization problem can become arbitrarily complex
- e.g. implementing ERM for halfspaces in the non-separable case is computationally hard (chapter 9)
- For many interesting classes, it is infeasible to implement ERM
    - Solving the optimization problem takes forever

ERM can be hard.

- Depending on $\mathcal{H}$, the optimization problem can become arbitrarily complex
- e.g. implementing ERM for halfspaces in the non-separable case is computationally hard (chapter 9)
- For many interesting classes, it is infeasible to implement ERM
  - Solving the optimization problem takes forever

...so what can we do?

A first idea...

Idea: Use simpler hypothesis classes where ERM isn't hard.

A first idea...

Idea: Use simpler hypothesis classes where ERM isn't hard.

- Problem: Simple classes can be too "weak" to estimate all relationships in the data
  - $\rightarrow$ Can lead to underfitting and poor performance

A first idea...

Idea: Use simpler hypothesis classes where ERM isn't hard.

- Problem: Simple classes can be too ”weak” to estimate all relationships in the data
  - $\rightarrow$ Can lead to underfitting and poor performance
- Approximation error is high ($\rightarrow$ B/C tradeoff)

A first idea...

Idea: Use simpler hypothesis classes where ERM isn't hard.

- Problem: Simple classes can be too "weak" to estimate all relationships in the data
  - $\rightarrow$ Can lead to underfitting and poor performance
- Approximation error is high ($\rightarrow$ B/C tradeoff)
- Still, these classes can be useful for us
  - If the resulting hypothesis is at least better than random

A first idea...

Idea: Use simpler hypothesis classes where ERM isn't hard.

- Problem: Simple classes can be too "weak" to estimate all relationships in the data
  - $\rightarrow$ Can lead to underfitting and poor performance
- Approximation error is high ($\rightarrow$ B/C tradeoff)
- Still, these classes can be useful for us
  - If the resulting hypothesis is at least better than random

Let's call ERM on a simple class a **weak learner**. We will formally define it later...

The idea behind boosting

The idea behind boosting

Why not combine many weak learners?
Can this give us an efficient strong learner?

The idea behind boosting

Why not combine many weak learners?
Can this give us an efficient strong learner?

- This theoretical question is the origin of boosting

The idea behind boosting

Why not combine many weak learners?
Can this give us an efficient strong learner?

- This theoretical question is the origin of boosting
- It was first raised in 1988 by Kearns and Valiant [2]

Why not combine many weak learners?
Can this give us an efficient strong learner?

- This theoretical question is the origin of boosting
- It was first raised in 1988 by Kearns and Valiant [2]
- The first (practical) answer was given in 1995 by Freund and Schapire [1]
  - $\rightarrow$ It is YES!

The idea behind boosting

Why not combine many weak learners?
Can this give us an efficient strong learner?

- This theoretical question is the origin of boosting
- It was first raised in 1988 by Kearns and Valiant [2]
- The first (practical) answer was given in 1995 by Freund and Schapire [1]
  - $\rightarrow$ It is YES!
- The result is **AdaBoost**, a widely popular and award winning algorithm
  - We will take a look at this later...

The idea behind boosting

Why not combine many weak learners?
Can this give us an efficient strong learner?

- This theoretical question is the origin of boosting
- It was first raised in 1988 by Kearns and Valiant [2]
- The first (practical) answer was given in 1995 by Freund and Schapire [1]
  - $\rightarrow$ It is YES!
- The result is **AdaBoost**, a widely popular and award winning algorithm
  - We will take a look at this later...

But first, let's get back to weak learning.

Weak Learnability

What exactly is a weak learner?

What exactly is a weak learner?

Remember, that a strong PAC learner for a class $\mathcal{H}$...

What exactly is a weak learner?

Remember, that a strong PAC learner for a class $\mathcal{H}$...

- ...if it is presented with $m > m_{\mathcal{H}}(\epsilon, \delta)$ examples

What exactly is a weak learner?

Remember, that a strong PAC learner for a class $\mathcal{H}$...

- ...if it is presented with $m > m_{\mathcal{H}}(\epsilon, \delta)$ examples
- ...has to find a hypothesis $h \in \mathcal{H}$

What exactly is a weak learner?

Remember, that a strong PAC learner for a class $\mathcal{H}$...

- ...if it is presented with $m > m_{\mathcal{H}}(\epsilon, \delta)$ examples
- ...has to find a hypothesis $h \in \mathcal{H}$
- ...such that $L_{(\mathcal{D},f)}(h) < \epsilon$ for every D and f with confidence $1 - \delta$ (if RA holds)

What exactly is a weak learner?

Remember, that a strong PAC learner for a class $\mathcal{H}$...

- ...if it is presented with $m > m_{\mathcal{H}}(\epsilon, \delta)$ examples
- ...has to find a hypothesis $h \in \mathcal{H}$
- ...such that $L_{(\mathcal{D},f)}(h) < \epsilon$ for every D and f with confidence $1 - \delta$ (if RA holds)

$\boxed{\text{In weak learning, we only want the error to be less than 50\%.}}$

Weak learning definition

An algorithm A is a γ-weak-learner for a class $\mathcal{H}$, if...

Weak learning definition

An algorithm A is a γ-weak-learner for a class $\mathcal{H}$, if...

- ...for every $\delta \in (0, 1)$ there exists a threshold $m_{\mathcal{H}}(\delta) \in \mathbb{N}$, such that

Weak learning definition

An algorithm A is a γ-weak-learner for a class $\mathcal{H}$, if...

- ...for every $\delta \in (0, 1)$ there exists a threshold $m_{\mathcal{H}}(\delta) \in \mathbb{N}$, such that
- ...if trained on at least $m > m_{\mathcal{H}}(\delta)$ examples

Weak learning definition

An algorithm A is a γ-weak-learner for a class $\mathcal{H}$, if...

- ...for every $\delta \in (0, 1)$ there exists a threshold $m_{\mathcal{H}}(\delta) \in \mathbb{N}$, such that
- ...if trained on at least $m > m_{\mathcal{H}}(\delta)$ examples
- ...it will find a hypothesis h, such that

Weak learning definition

An algorithm A is a γ-weak-learner for a class $\mathcal{H}$, if...

- ...for every $\delta \in (0, 1)$ there exists a threshold $m_{\mathcal{H}}(\delta) \in \mathbb{N}$, such that
- ...if trained on at least $m > m_{\mathcal{H}}(\delta)$ examples
- ...it will find a hypothesis h, such that
- ...$L_{(\mathcal{D},f)}(h) < \frac{1}{2} - \gamma$ with confidence $1 - \delta$

Weak learning definition

An algorithm A is a γ-weak-learner for a class $\mathcal{H}$, if...

- ...for every $\delta \in (0, 1)$ there exists a threshold $m_{\mathcal{H}}(\delta) \in \mathbb{N}$, such that
- ...if trained on at least $m > m_{\mathcal{H}}(\delta)$ examples
- ...it will find a hypothesis h, such that
- ...$L_{(\mathcal{D},f)}(h) < \frac{1}{2} - \gamma$ with confidence $1 - \delta$
- ...for every labeling function f and every distribution $\mathcal{D}$ (if RA holds)

Weak learning definition

An algorithm A is a γ-weak-learner for a class $\mathcal{H}$, if...

- ...for every $\delta \in (0, 1)$ there exists a threshold $m_{\mathcal{H}}(\delta) \in \mathbb{N}$, such that
- ...if trained on at least $m > m_{\mathcal{H}}(\delta)$ examples
- ...it will find a hypothesis h, such that
- ...$L_{(\mathcal{D},f)}(h) < \frac{1}{2} - \gamma$ with confidence $1 - \delta$
- ...for every labeling function f and every distribution $\mathcal{D}$ (if RA holds)

Lets look at an example (Decision Stumps)

Spam detection with decision stumps

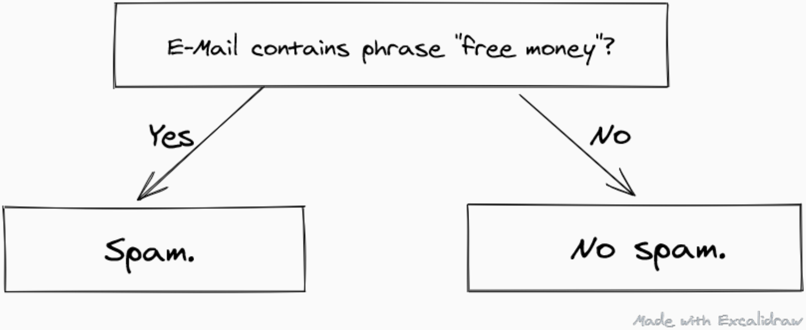


Figure 1: This is a Decision Stump.

ERM for decision stumps is efficient

[1] D_i are sample weights

ERM for decision stumps is efficient

- Decision Stumps partition the instance space $\mathcal{X}$ along a single dimension

[1] D_i are sample weights

ERM for decision stumps is efficient

- Decision Stumps partition the instance space $\mathcal{X}$ along a single dimension
- This is the hypothesis class:

[1] D_i are sample weights

ERM for decision stumps is efficient

- Decision Stumps partition the instance space $\mathcal{X}$ along a single dimension
- This is the hypothesis class:

$$\mathcal{H}_{DS} = \{\mathbf{x} \mapsto \operatorname{sign}(\theta - x_i) \cdot b : \ \theta \in \mathbb{R}, i \in [d], b \in \{\pm 1\}\}$$

[1] D_i are sample weights

ERM for decision stumps is efficient

- Decision Stumps partition the instance space $\mathcal{X}$ along a single dimension
- This is the hypothesis class:

$$\mathcal{H}_{DS} = \{\mathbf{x} \mapsto \operatorname{sign}(\theta - x_i) \cdot b : \ \theta \in \mathbb{R}, i \in [d], b \in \{\pm 1\}\}$$

- ERM has to find the best threshold θ and the best dimension $i \in [d]$ such that the training error is minimized:[1]

[1] D_i are sample weights

ERM for decision stumps is efficient

- Decision Stumps partition the instance space $\mathcal{X}$ along a single dimension
- This is the hypothesis class:

$$\mathcal{H}_{DS} = \{\mathbf{x} \mapsto \operatorname{sign}(\theta - x_i) \cdot b : \ \theta \in \mathbb{R}, i \in [d], b \in \{\pm 1\}\}$$

- ERM has to find the best threshold θ and the best dimension $i \in [d]$ such that the training error is minimized:[1]

$$\min_{j \in [d]} \min_{\theta \in \mathbb{R}} \left(\sum_{i: y_i = 1}^{m} D_i \mathbb{1}_{[x_{i,j} > \theta]} + \sum_{i: y_i = -1}^{m} D_i \mathbb{1}_{[x_{i,j} \le \theta]} \right)$$

[1] D_i are sample weights

ERM for decision stumps is efficient

- Decision Stumps partition the instance space $\mathcal{X}$ along a single dimension
- This is the hypothesis class:

$$\mathcal{H}_{DS} = \{\mathbf{x} \mapsto \mathrm{sign}\left(\theta - x_i\right) \cdot b : \ \theta \in \mathbb{R}, i \in [d], b \in \{\pm 1\}\}$$

- ERM has to find the best threshold θ and the best dimension $i \in [d]$ such that the training error is minimized:[1]

$$\min_{j \in [d]} \min_{\theta \in \mathbb{R}} \left(\sum_{i: y_i = 1}^{m} D_i \mathbb{1}_{[x_{i,j} > \theta]} + \sum_{i: y_i = -1}^{m} D_i \mathbb{1}_{[x_{i,j} \leq \theta]} \right)$$

$$\boxed{\text{This can be solved in } \mathcal{O}(dm)\text{!}}$$

[1] D_i are sample weights

Wisdom of the crowd

Wisdom of the crowd

So now we know what a weak learner is.
But how can weak learners be used to build something powerful?

Wisdom of the crowd

So now we know what a weak learner is.
But how can weak learners be used to build something powerful?

AdaBoost

AdaBoost - How it works

AdaBoost - How it works

- AdaBoost invokes the weak learner T times on the training data using different sample weights $(D_1^{(t)}, \ldots, D_m^{(t)}),\ t = 1, \ldots, T$

AdaBoost - How it works

- AdaBoost invokes the weak learner T times on the training data using different sample weights $(D_1^{(t)}, ..., D_m^{(t)}),\ t = 1, ..., T$
- In every iteration t, the weak learner finds a hypothesis h_t

AdaBoost - How it works

- AdaBoost invokes the weak learner T times on the training data using different sample weights $(D_1^{(t)}, ..., D_m^{(t)}),\ t = 1, ..., T$
- In every iteration t, the weak learner finds a hypothesis h_t
- Then, AdaBoost does a weighted majority vote:

$$h(x) = \text{sign}\left(\sum_{t=1}^{T} w_t h_t(x)\right)$$

AdaBoost - How it works

- AdaBoost invokes the weak learner T times on the training data using different sample weights $(D_1^{(t)}, \ldots, D_m^{(t)}),\ t = 1, \ldots, T$
- In every iteration t, the weak learner finds a hypothesis h_t
- Then, AdaBoost does a weighted majority vote:

$$h(x) = \operatorname{sign}\left(\sum_{t=1}^{T} w_t h_t(x)\right)$$

Let's look at this in more detail...

One iteration of AdaBoost

One iteration of AdaBoost

1. Invoke the weak learner on the training data weighted by $D^{(t)}$
   - In iteration $t = 1$, we use equal weights $D_i^{(t)} = \frac{1}{m}$

One iteration of AdaBoost

1. Invoke the weak learner on the training data weighted by $D^{(t)}$
   - In iteration $t = 1$, we use equal weights $D_i^{(t)} = \frac{1}{m}$
2. Compute a weight for the resulting hypothesis h_t like this:

$$w_t = \frac{1}{2}\log\left(\frac{1}{\epsilon_t} - 1\right)$$

   - ϵ_t is the (weighted) training error of h_t

One iteration of AdaBoost

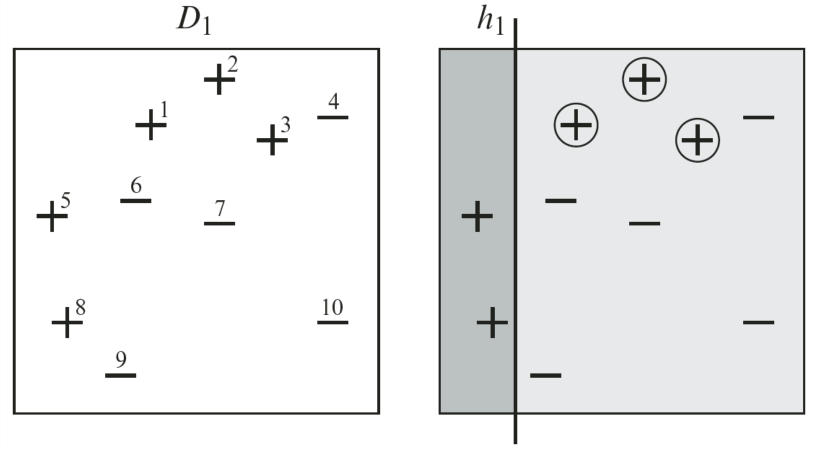
1. Invoke the weak learner on the training data weighted by $D^{(t)}$
   - In iteration $t = 1$, we use equal weights $D_i^{(t)} = \frac{1}{m}$
2. Compute a weight for the resulting hypothesis h_t like this:

$$w_t = \frac{1}{2}\log\left(\frac{1}{\epsilon_t} - 1\right)$$

   - ϵ_t is the (weighted) training error of h_t
3. Update the weights $D_i^{(t)}$ like this

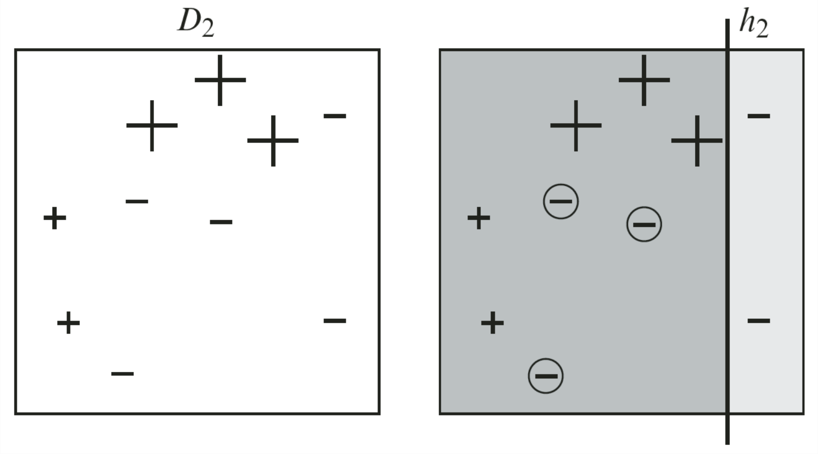
$$D_i^{(t+1)} = \frac{D_i^{(t)}\exp\left(-w_t y_i h_t(\mathbf{x}_i)\right)}{\sum_{j=1}^{m} D_j^{(t)}\exp\left(-w_t y_j h_t(\mathbf{x}_j)\right)}$$

A step by step example[2]

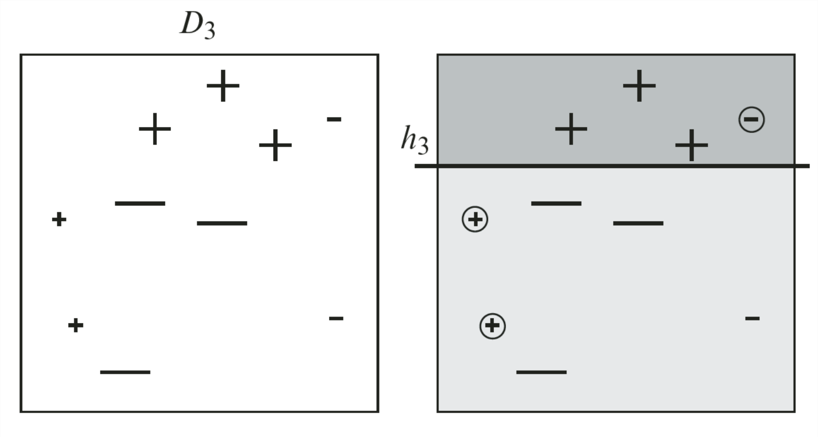


[2]Taken from the book *Boosting: Foundations and Algorithms* written by Freund and Schapire [3]. You can read it for free at `https://mitpress.mit.edu/books/boosting`

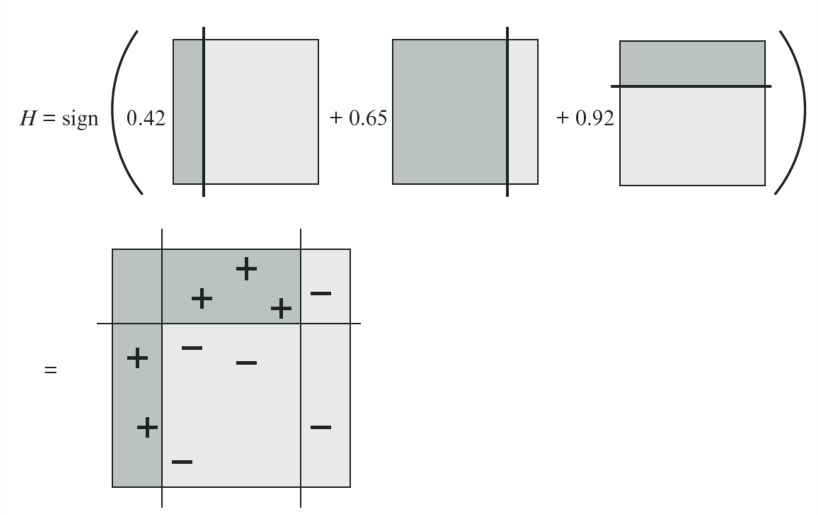
A step by step example



A step by step example



A step by step example



AdaBoost training error

AdaBoost training error

- In the previous example, the training error was reduced to zero

AdaBoost training error

- In the previous example, the training error was reduced to zero
- What about the general case?

AdaBoost training error

- In the previous example, the training error was reduced to zero
- What about the general case?

$$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{[h(\mathbf{x}_i) \neq y_i]} \leq e^{-2\gamma^2 T}$$

AdaBoost training error

- In the previous example, the training error was reduced to zero
- What about the general case?

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{[h(\mathbf{x}_i) \neq y_i]} \leq e^{-2\gamma^2 T}$$

- The training error of AdaBoost decreases exponentially in T

AdaBoost training error

- In the previous example, the training error was reduced to zero
- What about the general case?

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{[h(\mathrm{x}_i)\neq y_i]} \leq e^{-2\gamma^2 T}$$

- The training error of AdaBoost decreases exponentially in T

...but what about the out of sample error?

Conclusion

Conclusion

The End.

Y. Freund and R. E. Schapire.
A decision-theoretic generalization of on-line learning and an application to boosting.
Journal of Computer and System Sciences, 55(1):119 – 139, 1997.

M. Kearns and L. G. Valiant.
Learning boolean formulae or finite automata is as hard as factoring.
Technical Report TR 14-88, Harvard University Aiken Computation Laboratory, 1988.

R. E. Schapire and Y. Freund.
Boosting: Foundations and Algorithms.
The MIT Press, 2012.

S. Shalev-Shwartz and S. Ben-David.
Understanding Machine Learning - From Theory to Algorithms.
Cambridge University Press, 2014.