Introduction to Boosting

February 2, 2012

Outline

Boosting - Introduction

- Train one base learner at a time.
- Focus it on the mistakes of its predecessors.
- Weight it based on how 'useful' it is in the ensemble (*not* on its training error).

Boosting - Introduction

Under very mild assumptions on the base learners ("Weak learning" - they must perform better than random guessing):

1. Training error is eventually reduced to zero (We can fit the training data).
2. Generalisation error continues to be reduced even after training error is zero (There is no overfitting).

Boosting - Introduction

- The most popular ensemble algorithm is a boosting algorithm called “Adaboost”.
- Adaboost is short for “Adaptive Boosting”, because the algorithm adapts weights on the base learners and training examples.
- There are *many* explanation of precisely what Adaboost does and why it is so successful - but the basic idea is simple!

Outline

Adaboost - Pseudo-code

Adaboost Pseudo-code

```
Set uniform example weights.
for Each base learner do
    Train base learner with weighted sample.
    Test base learner on all data.
    Set learner weight with weighted error.
    Set example weights based on ensemble predictions.
end for
```

Adaboost - Loss Function

To see precisely how Adaboost is implemented, we need to understand its *Loss Function*

$$\mathcal{L}(H) = \sum_{i=1}^{N} e^{-m_i} \tag{1}$$

$$m_i = y_i \sum_{k=1}^{K} \alpha_k h_k(x_i) \tag{2}$$

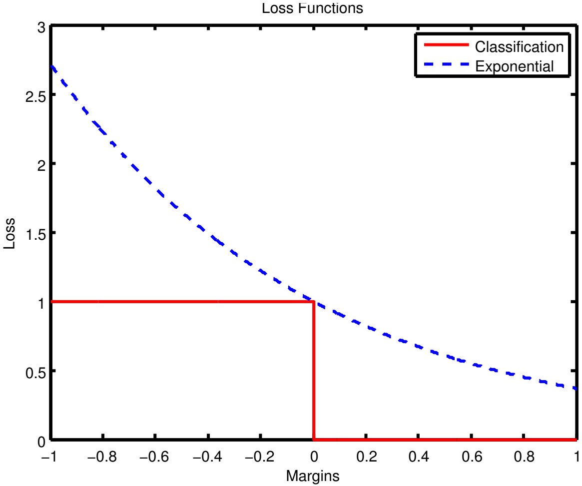
m_i is the *voting* margin.
There are N examples and K base learners.
α_K is the weight of the k^{th} learner, $h_k(x_i)$ is its prediction on x_i.

Adaboost - Loss Function

Why Exponential Loss?

- Loss is higher when a prediction is wrong.
- Loss is *steeper* when a prediction is wrong.
- Precise reasons later...

Adaboost - Loss Function



Adaboost - Example Weights

After k learners, the example weights used to train the $k+1^{\text{th}}$ learner are:

$$\begin{aligned} D_k(i) &\propto e^{-m_i} & (3)\\ &= e^{y_i \sum_{j=1}^{k} \alpha_j h_j(x_i)} & (4)\\ &= e^{y_i \sum_{j=1}^{k-1} \alpha_j h_j(x_i)} e^{y_i \alpha_k h_k(x_i)} & (5)\\ &\propto D_{k-1}(i) e^{y_i \alpha_k h_k(x_i)} & (6)\\ D_k(i) &= \frac{D_{k-1}(i) e^{y_i \alpha_k h_k(x_i)}}{Z_k} & (7) \end{aligned}$$

- Z_k normalises $D_k(i)$ such that $\sum_{i=1}^{N} D_k(i) = 1$.
- Larger when m_i is negative.
- Directly proportional to the loss funcion.

Adaboost - Learner Weights

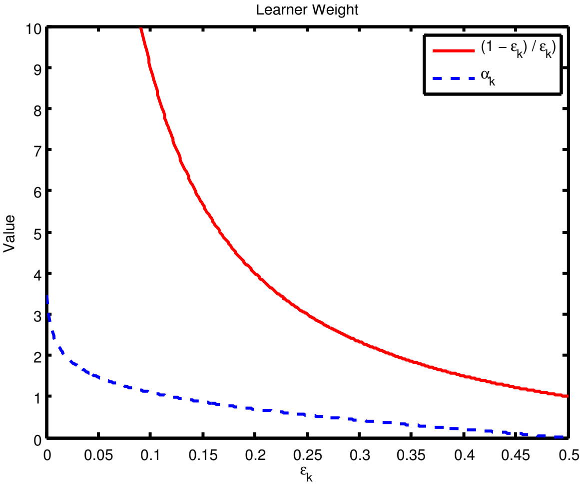
Once the k^{th} learner is trained, we evaluate its predictions on training data $h_k(x_i)$ and assign weight:

$$\alpha_k = \frac{1}{2} \log \frac{1 - \epsilon_k}{\epsilon_k}, \tag{8}$$

$$\epsilon_k = \sum_{i=1}^{N} D_{k-1}(i) \delta[h_k(x_i) \neq y_i] \tag{9}$$

- ϵ_k is the *weighted error* of the k^{th} learner.
- The log in the learner weight is related to the exponential in the loss function.

Adaboost - Learner Weights



Adaboost - Pseudo-code

Adaboost Pseudo-code

$D_k(i)$: Example i weight after learner k
α_k: Learner k weight
Set uniform example weights.
for Each base learner **do**
  Train base learner with weighted sample.

  Test base learner on all data.
  Set learner weight with weighted error.
  Set example weights based on ensemble predictions.
end for

Adaboost - Pseudo-code

Adaboost Pseudo-code

$D_k(i)$: Example i weight after learner k

α_k: Learner k weight

$\forall i : D_0(i) \leftarrow \frac{1}{N}$

for Each base learner **do**

    Train base learner with weighted sample.

    Test base learner on all data.

    Set learner weight with weighted error.

    Set example weights based on ensemble predictions.

end for

Adaboost - Pseudo-code

Adaboost Pseudo-code

$D_k(i)$: Example i weight after learner k
α_k: Learner k weight
$\forall i : D_0(i) \leftarrow \frac{1}{N}$
for k=1 to K **do**
  $\mathcal{D} \leftarrow$ data sampled with D_{k-1}.
  $h_k \leftarrow$ base learner trained on $\mathcal{D}$
  Test base learner on all data.
  Set learner weight with weighted error.
  Set example weights based on ensemble predictions.
end for

Adaboost - Pseudo-code

Adaboost Pseudo-code

$D_k(i)$: Example i weight after learner k
α_k: Learner k weight
$\forall i : D_0(i) \leftarrow \frac{1}{N}$
for k=1 to K **do**
  $\mathcal{D} \leftarrow$ data sampled with D_{k-1}.
  $h_k \leftarrow$ base learner trained on $\mathcal{D}$
  $\epsilon_k \leftarrow \sum_{i=1}^{N} D_{k-1}(i)\delta[h_k(x_i) \neq y_i]$
  Set learner weight with weighted error.
  Set example weights based on ensemble predictions.
end for

Adaboost - Pseudo-code

Adaboost Pseudo-code

$D_k(i)$: Example i weight after learner k
α_k: Learner k weight
$\forall i : D_0(i) \leftarrow \frac{1}{N}$
for k=1 to K **do**
  $\mathcal{D} \leftarrow$ data sampled with D_{k-1}.
  $h_k \leftarrow$ base learner trained on $\mathcal{D}$
  $\epsilon_k \leftarrow \sum_{i=1}^{N} D_{k-1}(i)\delta[h_k(x_i) \neq y_i]$
  $\alpha_k = \frac{1}{2} \log \frac{1-\epsilon_k}{\epsilon_k}$
  Set example weights based on ensemble predictions.
end for

Adaboost - Pseudo-code

Adaboost Pseudo-code

$D_k(i)$: Example i weight after learner k

α_k: Learner k weight

$\forall i : D_0(i) \leftarrow \frac{1}{N}$

for k=1 to K **do**

  $\mathcal{D} \leftarrow$ data sampled with D_{k-1}.

  $h_k \leftarrow$ base learner trained on $\mathcal{D}$

  $\epsilon_k \leftarrow \sum_{i=1}^{N} D_{k-1}(i)\delta[h_k(x_i) \neq y_i]$

  $\alpha_k \leftarrow \frac{1}{2} \log \frac{1-\epsilon_k}{\epsilon_k}$

  $D_k(i) \leftarrow \frac{D_{k-1}(i)e^{-\alpha_k y_i h_k(x_i)}}{Z_k}$

end for

Adaboost - Pseudo-code

Decision rule:

$$H(x) = sign(\sum_{k=1}^{K} \alpha_k h_k(x)) \tag{10}$$

Outline

Adaboost - Demo

[Demo]

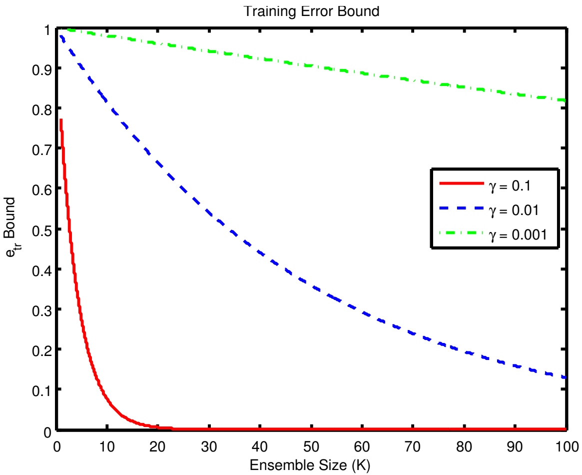
Outline

Adaboost - Training Error

- So the training error rate of Adaboost reaches 0 for large enough K
- Combination of exponential loss + weak learning assumption.
- If base learner weighted error is always $\epsilon \leq \frac{1}{2} - \gamma$ then:

$$e_{\text{tr}} \leq (\sqrt{1 - 4\gamma^2})^K \tag{11}$$

Adaboost - Training Error



Adaboost - Training Error

Let's derive the bound!

1. Show that exponential loss is an upper bound on classification error.
2. Show that the loss can be written as a product of the normalisation constant.
3. Minimise the normalisation constant at each step (derive the learner weight rule).
4. Write this constant using the weighted error at each step.

Adaboost - Training Error

Show that exponential loss is an upper bound on classification error.

$$e_{\text{tr}} = \frac{1}{N}\sum_{i=1}^{N}\delta[H(x_i) \neq y_i] \quad (12)$$

$$= \frac{1}{N}\sum_{i=1}^{N}\delta[m_i \leq 0] \quad (13)$$

$$\leq \frac{1}{N}\sum_{i=1}^{N} e^{-m_i} \quad (14)$$

Because $x \leq 0 \rightarrow e^{-x} \geq 1$ and $x \leq 1 \rightarrow e^{-x} \geq 0$

Adaboost - Training Error

Show that the loss can be written as a product of the normalisation constant.

$$
\begin{aligned}
D_k(i) &= \frac{D_{k-1}(i)e^{-y_i \alpha_k h_k(x_i)}}{Z_k} && (15)\\
&= D_0(i)[\prod_{j=1}^{k} \frac{e^{-y_i \alpha_j h_j(x_i)}}{Z_j}] && (16)\\
&= \frac{1}{N} e^{-y_i \sum_{j=1}^{k} \alpha_j h_j(x_i)} \frac{1}{\prod_{j=1}^{k} Z_j} && (17)\\
e^{-m_i} &= N D_k(i) \prod_{j=1}^{k} Z_j && (18)
\end{aligned}
$$

Adaboost - Training Error

Show that the loss can be written as a product of the normalisation constant.

$$e^{-m_i} = ND_k(i)\prod_{j=1}^{k} Z_j \tag{19}$$

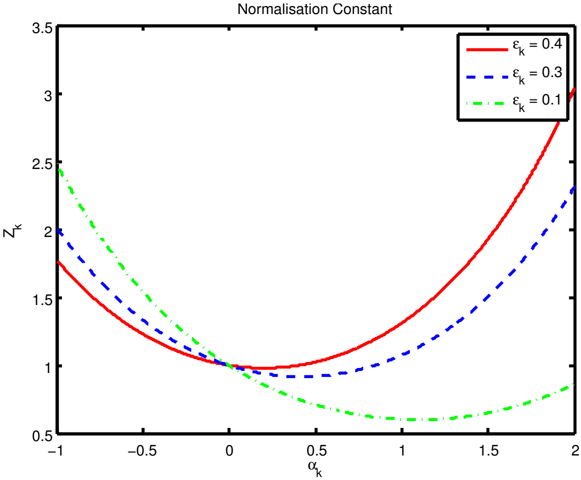
$$\begin{aligned} e_{\text{tr}} &\leq \frac{1}{N}\sum_{i=1}^{N} e^{-m_i} && (20)\\ &= \frac{1}{N}\sum_{i=1}^{N} ND_k(i)\prod_{j=1}^{k} Z_j && (21)\\ &= [\sum_{i=1}^{N} D_k(i)]\prod_{j=1}^{k} Z_j && (22)\\ &= \prod_{j=1}^{k} Z_j && (23)\end{aligned}$$

Adaboost - Training Error

Minimise normalisation constant at each step.

$$\begin{aligned} Z_k &= \sum_{i=1}^{N} D_{k-1}(i) e^{-y_i \alpha_k h_k(x_i)} && (24) \\ &= \sum_{h_k(x_i) \neq y_i} D_{k-1}(i) e^{\alpha_k} + \sum_{h_k(x_i) = y_i} D_{k-1}(i) e^{-\alpha_k} && (25) \\ &= \epsilon_k e^{\alpha_k} + (1 - \epsilon_k) e^{-\alpha_k} && (26) \end{aligned}$$

Adaboost - Training Error



Adaboost - Training Error

Minimise normalisation constant at each step.

$$
\begin{aligned}
Z_k &= \epsilon_k e^{\alpha_k} + (1-\epsilon_k)e^{-\alpha_k} && (27)\\
\frac{\partial Z_k}{\partial \alpha_k} &= \epsilon_k e^{\alpha_k} - (1-\epsilon_k)e^{-\alpha_k} && (28)\\
0 &= \epsilon_k e^{2\alpha_k} - (1-\epsilon_k)e^{0} && (29)\\
e^{2\alpha_k} &= \frac{1-\epsilon_k}{\epsilon_k} && (30)\\
\alpha_k &= \frac{1}{2}\log\frac{1-\epsilon_k}{\epsilon_k} && (31)
\end{aligned}
$$

Minimising Z_k gives us the learner weight rule.

Adaboost - Training Error

Write the normalisation constant using weighted error.

$$
\begin{aligned}
Z_k &= \epsilon_k e^{\alpha_k} + (1-\epsilon_k)e^{-\alpha_k} && (32)\\
\alpha_k &= \frac{1}{2}\log\frac{1-\epsilon_k}{\epsilon_k} && (33)\\
Z_k &= \epsilon_k e^{\frac{1}{2}\log\frac{1-\epsilon_k}{\epsilon_k}} + (1-\epsilon_k)e^{-\frac{1}{2}\log\frac{1-\epsilon_k}{\epsilon_k}} && (34)\\
&= 2\sqrt{\epsilon_k(1-\epsilon_k)} && (35)
\end{aligned}
$$

So Z_k depends only on ϵ_k, and the weak learning assumption says that $\epsilon_k < 0.5$.

Adaboost - Training Error

- We bound training error using: $e_{\text{tr}} \leq \prod_{j=1}^{k} Z_j$
- We've found that Z_j is a function of the weighted error: $Z_j = 2\sqrt{\epsilon_j(1-\epsilon_j)}$
- So, we can bound the training error: $e_{\text{tr}} \leq \prod_{j=1}^{k} 2\sqrt{\epsilon_j(1-\epsilon_j)}$
- Let $\epsilon_j \leq \frac{1}{2} - \gamma$, and we can write $e_{\text{tr}} \leq (\sqrt{1-4\gamma^2})^k$
- If $\gamma > 0$, then this bound will shrink to 0 when k is large enough.

Adaboost - Training Error Summary

- We have seen an upper bound on the training error which decreases as K increase.
- We can prove the bound by considering the loss function as a bound on the error rate.
- Examining the role of the normalisation constant in this bound reveals the learner weight rule.
- Examining the role of γ in the bound shows why we can achieve 0 training error even with weak learners.