CS 4375 – Introduction to Machine Learning

Ensemble Methods – Boosting

Erick Parolin

THE UNIVERSITY OF TEXAS AT DALLAS

[Based on the slides of Nicholas Ruozzi, Based on the slides of Vibhav Gogate and Rob Schapire]

Previously

- **Variance reduction via bagging**
  - Generate “new” training data sets by sampling with replacement from the empirical distribution
  - Learn a classifier for each of the newly sampled sets
  - Combine the classifiers for prediction
- **How about reducing bias for binary classification problems?**

Boosting

- How to translate ***rules of thumb*** into good learning algorithms?
- **Examples**:
  - In medical diagnosis, classifying patients as either at risk or not at risk for developing diabetes.
    - If the patient's body mass index (BMI) is above 30, classify them as at risk.
    - If the patient’s age is over 45, classify them as at risk.
  - To classify emails as spam or not spam.
    - If the word "discount" appears in the email, classify it as spam.
    - If the sender’s email address is unknown, classify it as spam.
  - To detect credit card fraud transactions
    - If the transaction amount is unusually high compared to the user’s average spending, classify it as fraudulent.
    - If the transaction is made in a location far from the user’s usual area, classify it as fraudulent.

Boosting

- Freund & Schapire: Theory for "**weak learners**" in late 80's
- **Weak Learner:** performance on any training set is slightly better than chance prediction
- Intended to answer a theoretical question, not as a practical way to improve learning
  - Tested in mid 90's using *not-so-weak* learners
  - Works pretty well in practice!

Boosting

- **Main idea:** use weak learner to create strong learner.
- **Ensemble method:** combine base classifiers returned by weak learner.
- Finding simple relatively accurate base classifiers often not hard.
- **But, how should base classifiers be combined?**

Boosting

- Instead of learning a single (weak) classifier, learn many weak classifiers that are **good at different parts of the input space**.
- Output class: (Weighted) vote of each classifier
  - Classifiers that are most confident will vote with more conviction
  - Classifiers will be most confident about a particular part of the space
  - On average, it does better than single classifier!
    - **Force** classifiers to learn about different parts of the input space
    - Weight the votes of different classifiers

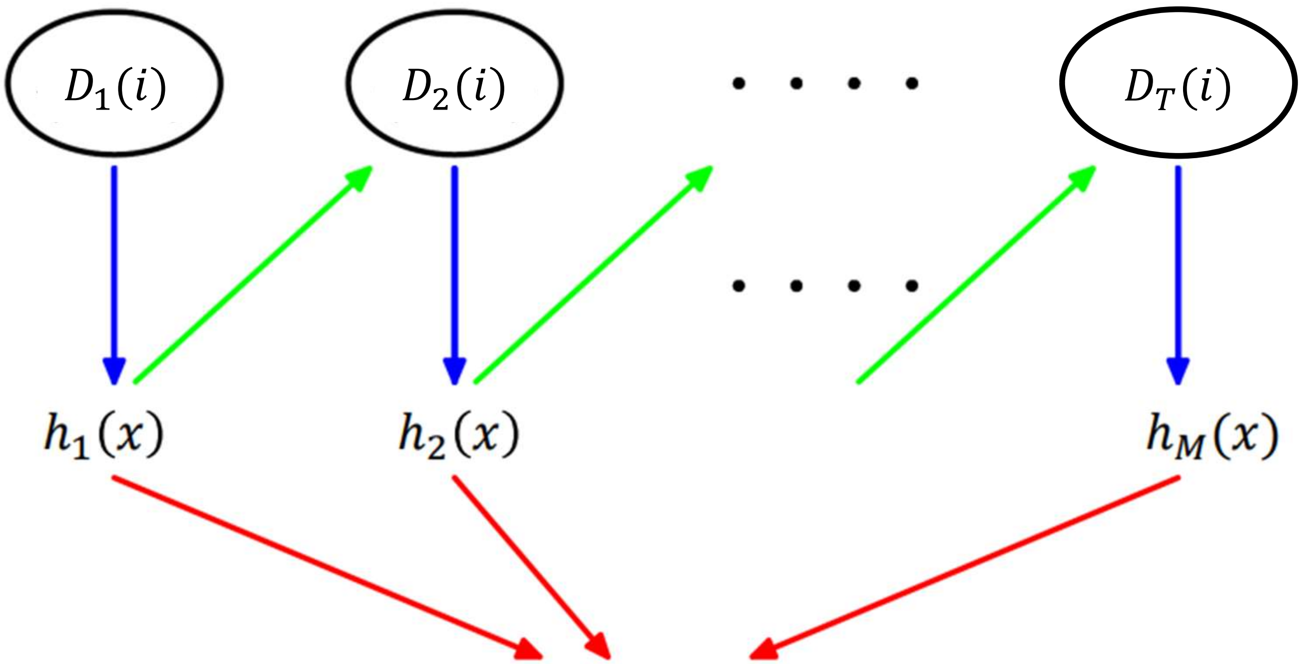
Boosting – General Algorithm

- Weight all training samples equally
- Train model on training set
- Compute **error** of model on training set
- **Increase weights** on training cases model **gets wrong**
- Train new model on **re-weighted** training set
- Re-compute **errors** on weighted training set
- **Increase weights** again on cases model **gets wrong**

Repeat until tired (100+ iterations)

Final model: weighted prediction of each model

Boosting – Graphical Illustration



D_t is a distribution that defines the weights for datapoints on round t.

Weak learner h_t will learn based on D_t, which is updated based on the training errors observed from h_{t-1}

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Final hypothesis H is composed by the combination of votes from each weak hypothesis (where α_t corresponds to the weight associated to hypothesis h_t).

Adaboost

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X$, $y_i \in \{-1, +1\}$
Initialize: $D_1(i) = 1/m \; for \; i = 1, \dots, m$

For t = 1,…,T:
Train weak learner using distribution D_t
Get weak hypothesis $h_t\colon X \to \{-1, +1\}$
Aim: select h_t with low weighted error:

$$\varepsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$

Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

Update, for $i = 1, \dots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t}$$

where Z_t is a normalization factor so that D_{t+1} sums to 1 (D_{t+1} will be a distribution)

Output the final hypothesis: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

Adaboost

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$
Initialize: $D_1(i) = 1/m \; for \; i = 1, \dots, m$

$\boldsymbol{D_t(i)}$ corresponds to the weight associated to i^{th} datapoint on t^{th} round/weak learner

For t = 1,…,T:

Train weak learner using distribution D_t

Get weak hypothesis h_t: $X \rightarrow \{-1, +1\}$

<u>Aim</u>: select h_t with low weighted error:

$$\varepsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$

Weighted number of incorrect classifications of the t^{th} classifier.

$$\varepsilon_t = \frac{\sum_i^m D_t(i) \, I(h_t(x_i) \neq y_i)}{\sum_i^m D_t(i)}$$

where I(·) is an indicator function (1 if incorrect, 0 if correct).

Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$

Update, for $i = 1, \dots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t}$$

where Z_t is a normalization factor so that D_{t+1} sums to 1 (D_{t+1} will be a distribution)

Output the final hypothesis: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

Adaboost

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in \{-1, +1\}$
Initialize: $D_1(i) = 1/m \; for \; i = 1, \ldots, m$

For t = 1,…,T:

Train weak learner using distribution D_t

Get weak hypothesis $h_t \colon X \rightarrow \{-1, +1\}$

Aim: select h_t with low weighted error:

$$\varepsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$

Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t}$$

where Z_t is a normalization factor so that D_{t+1} sums to 1 (D_{t+1} will be a distribution)

Output the final hypothesis: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

α_t reflects the confidence in each weak learner t, better-performing learners get more weight, while worse-performing learners contribute less.

$$\varepsilon_t \rightarrow 0 \Rightarrow \alpha_t \rightarrow \infty$$
$$\varepsilon_t \rightarrow 0.5 \Rightarrow \alpha_t \rightarrow 0$$
$$\varepsilon_t \rightarrow 1 \Rightarrow \alpha_t \rightarrow -\infty$$

Note: If $\varepsilon_t > 50\%$, the weights are reverted back to $1/m$, and the resampling procedure is repeated.

Adaboost

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X$, $y_i \in \{-1, +1\}$
Initialize: $D_1(i) = 1/m \; for \; i = 1, \dots, m$

For t = 1,…,T:

Train weak learner using distribution D_t

Get weak hypothesis h_t: $X \rightarrow \{-1, +1\}$

Aim: select h_t with low weighted error:

$$\varepsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$

Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$

Update, for $i = 1, \dots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t}$$

where Z_t is a normalization factor so that D_{t+1} sums to 1 (D_{t+1} will be a distribution)

Output the final hypothesis: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

Updating the weights: if an example was misclassified by h_t, then $D_{t+1}(i)$ increases for the next round, making i[th] data point more important for the next weak learner.

$\exp\left(-\alpha_t y_i h_t(x_i)\right) = e^{-\alpha_t}$ if $y_i = h_t(x_i)$
$\exp\left(-\alpha_t y_i h_t(x_i)\right) = e^{\alpha_t}$ if $y_i \neq h_t(x_i)$

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)$$

Adaboost

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in \{-1, +1\}$
Initialize: $D_1(i) = 1/m \; for \; i = 1, \ldots, m$

For t = 1,…,T:

Train weak learner using distribution D_t
Get weak hypothesis $h_t\colon\ X \to \{-1, +1\}$
Aim: select h_t with low weighted error:

$$\varepsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$

Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t}$$

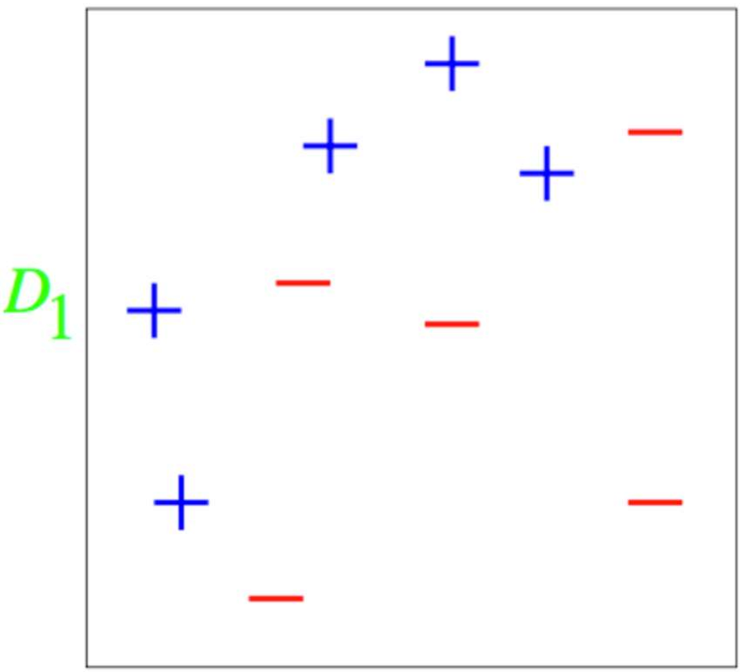
where Z_t is a normalization factor so that D_{t+1} sums to 1 (D_{t+1} will be a distribution)

Output the final hypothesis: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

Voted combinations of each weak hypothesis, where the votes α_t is used to emphasize component classifiers that are more reliable/confident than others.

Adaboost – Numerical Example

Let's say we have the following training data: 10 data points, 5 of each class.



Then, we have $\boldsymbol{m} = \mathbf{10}$, and following the algorithm, the initial weight distribution will be:

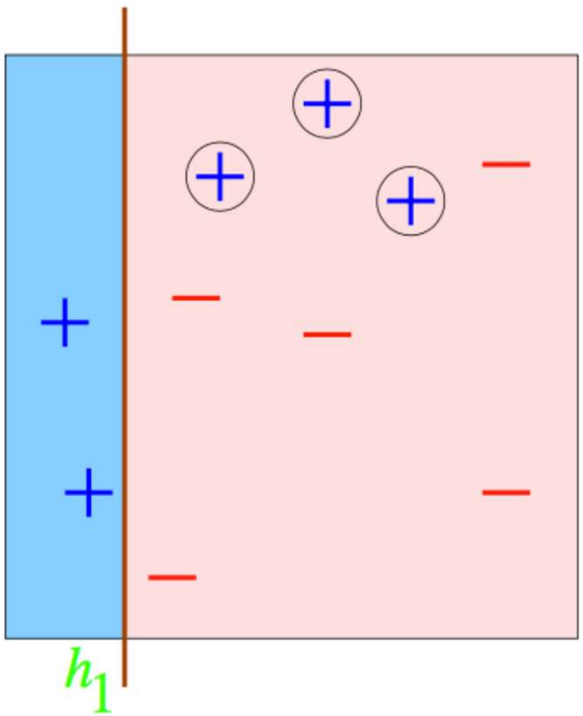
$D_1 = 1/10$

Which means:

$D_1(1) = 0.1, D_1(2) = 0.1, \ldots, D_1(10) = 0.1$

Adaboost – Numerical Example

Then, we train our first weak hypothesis h_1



3 positive points are misclassified

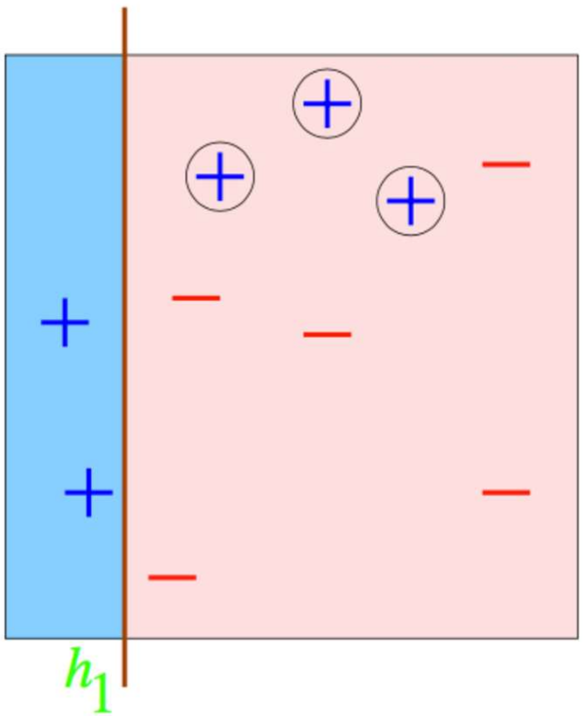
Then, we have:

$$\varepsilon_1 = \frac{\sum_{i=1}^{10} D_1(i)\, I(h_1(x_i) \neq y_i)}{\sum_{i=1}^{m} D_1(i)} = \frac{3 \times 0.1}{1} = 0.30$$

$$\alpha_1 = \frac{1}{2} ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right) = \frac{1}{2} ln\left(\frac{1-0.30}{0.30}\right) = 0.424$$

Adaboost – Numerical Example

Computing new weight distribution D_2



The weight points for the next round (round 2) is:

$$D_2(i) = \frac{D_1(i)\exp\big(-\alpha_1 y_i h_1(x_i)\big)}{\sum_{i=1}^{m} D_1(i)\exp\big(-\alpha_1 y_i h_1(x_i)\big)}$$

Misclassified points: $0.1 \times \exp(0.424) = 0.153$
Correctly classified points: $0.1 \times \exp(-0.424) = 0.065$

$$\sum_{i=1}^{m} D_1(i)\exp\big(-\alpha_1 y_i h_1(x_i)\big) = (3 \times 0.153) + (7 \times 0.065) = 0.914$$

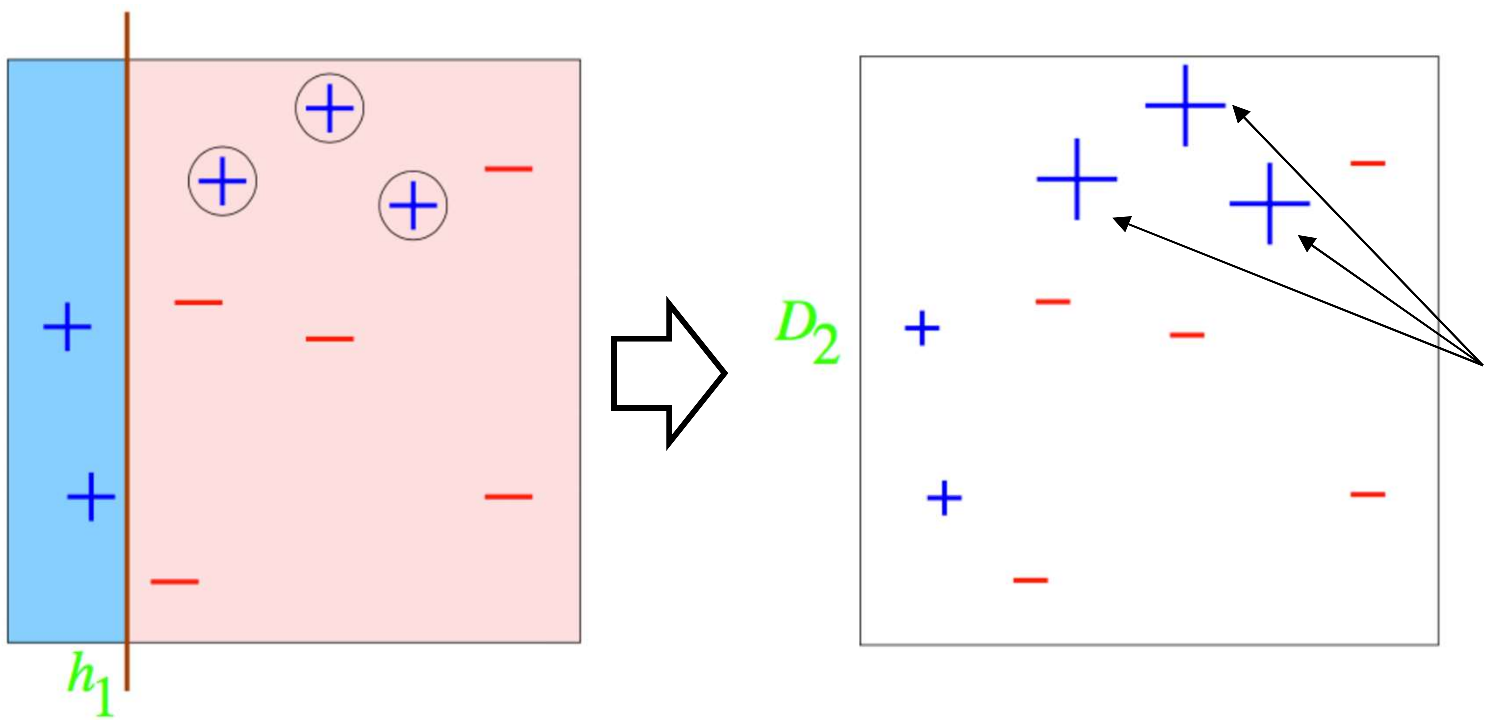
Normalizing, we have:

Misclassified points:$D_2(i) = \frac{0.1\times\exp(0.424)}{0.914} = 0.167$

Correctly classified points:$D_2(i) = \frac{0.1\times\exp(-0.424)}{0.914} = 0.071$

Adaboost – Numerical Example

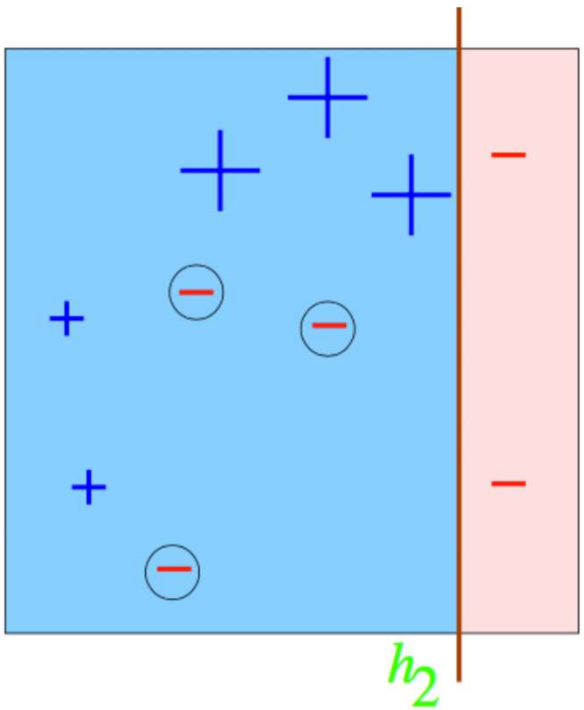
New weight distribution: After round 1, we get a new distribution D2



The 3 misclassified points get a higher weight

Adaboost – Numerical Example

We train the weak hypothesis h_2



3 positive points are misclassified

Then, we have:

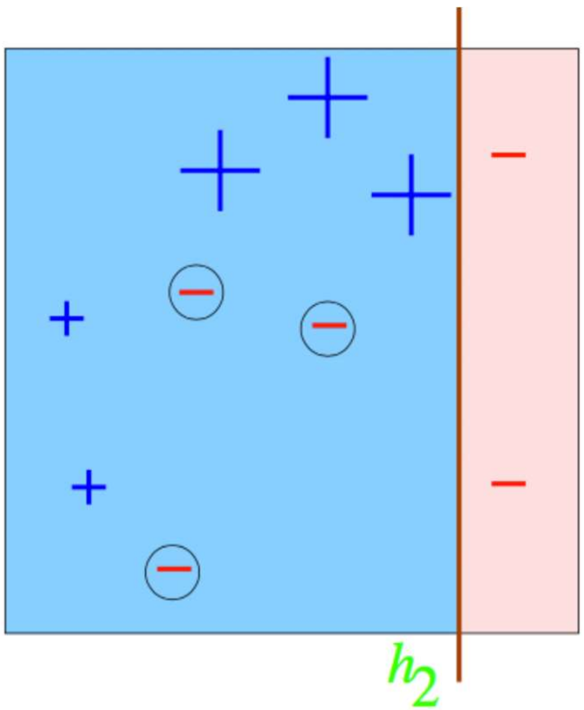
$$\varepsilon_2 = \frac{\sum_{i=1}^{10} D_2(i)\, I(h_2(x_i) \neq y_i)}{\sum_{i=1}^{m} D_2(i)} = \frac{3 \times 0.071}{1} = 0.213$$

$$\alpha_2 = \frac{1}{2} ln\left(\frac{1-\varepsilon_2}{\varepsilon_2}\right) = \frac{1}{2} ln\left(\frac{1-0.213}{0.213}\right) = 0.653$$

Note: we only used weight value $D_2(i) = 0.071$ to compute ε_2 because all the three misclassified points have the same weight (those points were not misclassified in round 1).
None of the points misclassified in round 1 were also misclassified in round 2.

Adaboost – Numerical Example

Computing new weight distribution D_3



The weight points for the next round (round 3) is:

$$D_3(i) = \frac{D_2(i)\exp\left(-\alpha_2 y_i h_2(x_i)\right)}{\sum_{i=1}^{m} D_2(i)\exp\left(-\alpha_2 y_i h_2(x_i)\right)}$$

Misclassified points: $D_2(i)\exp\left(-\alpha_2 y_i h_2(x_i)\right) = 0.071 \times \exp(0.653) = 0.136$

Correctly classified points:

Larger + $\Rightarrow D_2(i)\exp\left(-\alpha_2 y_i h_2(x_i)\right) = 0.167 \times \exp(-0.653) = 0.087$

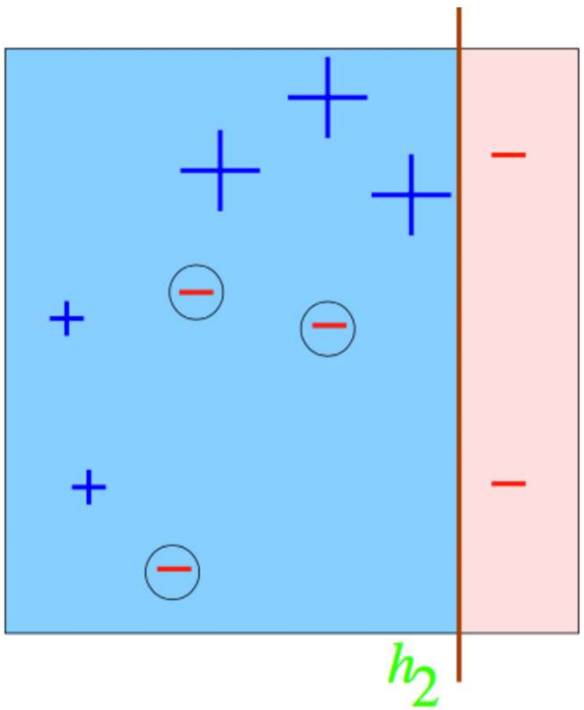
Smaller + and - $\Rightarrow D_2(i)\exp\left(-\alpha_2 y_i h_2(x_i)\right) = 0.071 \times \exp(-0.653) = 0.037$

Total Weight:

$$Z_t = \sum_{i=1}^{m} D_2(i)\exp\left(-\alpha_2 y_i h_2(x_i)\right) = 3 \times 0.136 + 3 \times 0.087 + 4 \times 0.037 = 0.817$$

Adaboost – Numerical Example

Computing new weight distribution D_3



Normalizing, we have:

Misclassified points: $\frac{0.136}{0.817} = 0.167$

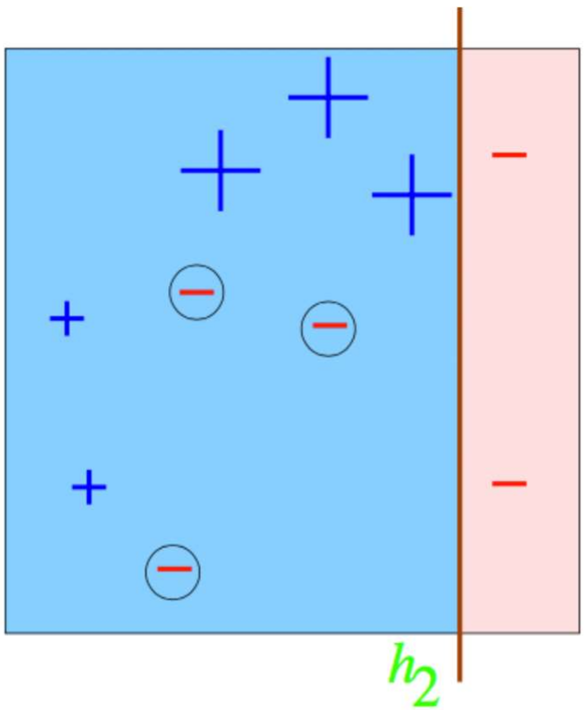
Correctly classified points:

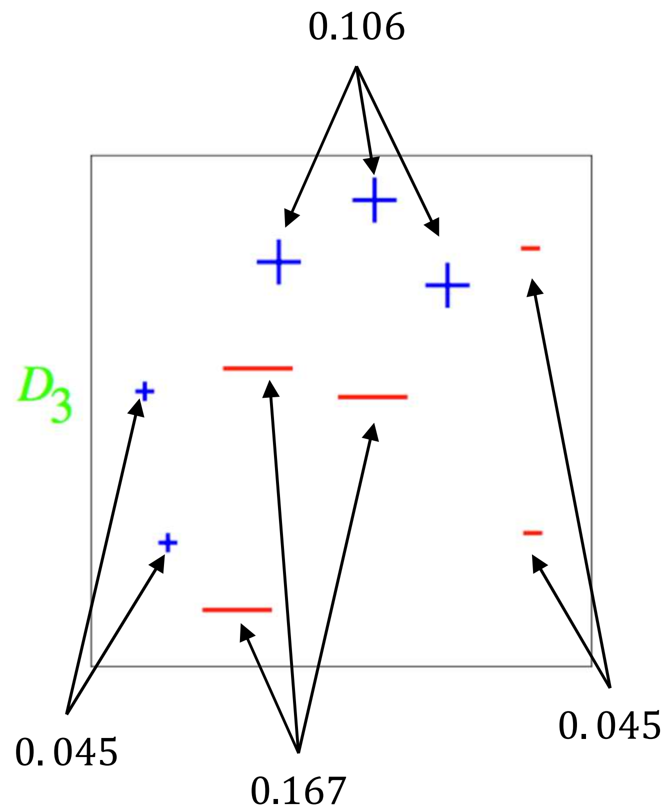
Larger **+** $\Rightarrow D_3(i) = \frac{0.087}{0.817} = 0.106$

Smaller + $\Rightarrow D_3(i) = \frac{0.037}{0.817} = 0.045$

Adaboost – Numerical Example

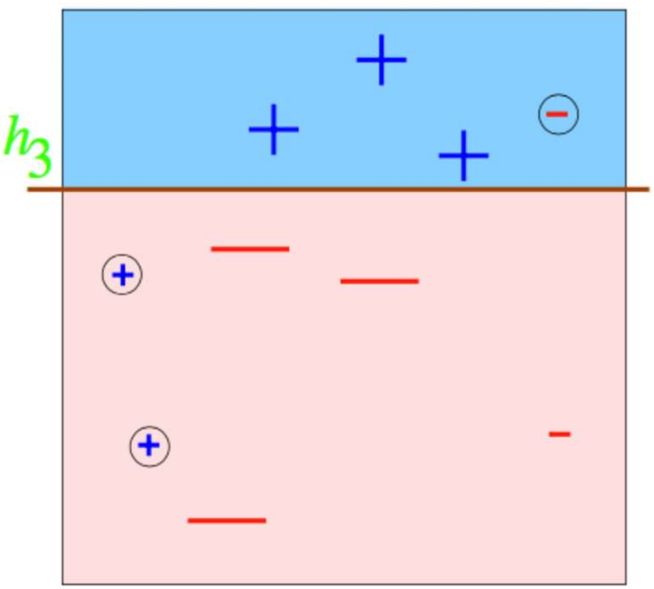
New weight distribution D_3





Adaboost – Numerical Example

Again, we train the weak hypothesis h_3



3 positive points are misclassified

Then, we have:

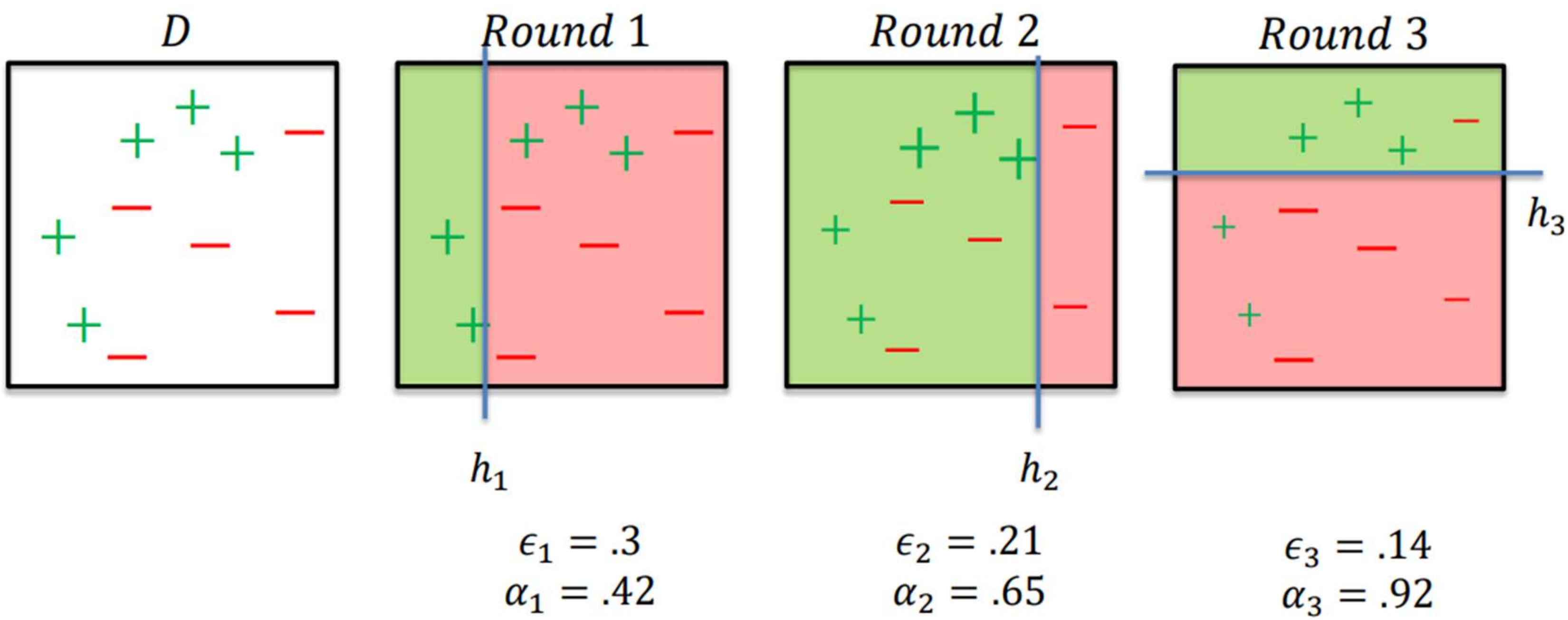
$$\varepsilon_3 = \frac{\sum_{i=1}^{10} D_3(i)\, I(h_2(x_i) \neq y_i)}{\sum_{i=1}^{m} D_3(i)} = \frac{3 \times 0.045}{1} = 0.135$$

$$\alpha_3 = \frac{1}{2} ln\left(\frac{1-\varepsilon_3}{\varepsilon_3}\right) = \frac{1}{2} ln\left(\frac{1-0.135}{0.135}\right) = 0.929$$

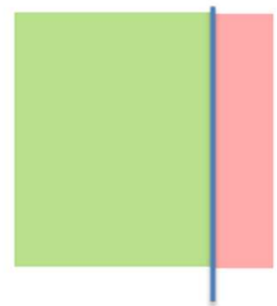
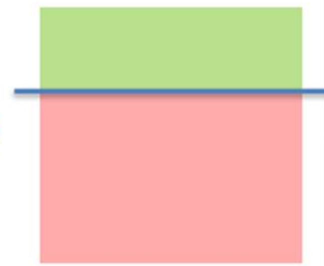
Note: again, we only used weight value $D_3(i) = 0.045$ to compute ε_3 because all the three misclassified points have the same weight (those points were not misclassified neither in round 1 nor in round 2).

Adaboost – Numerical Example

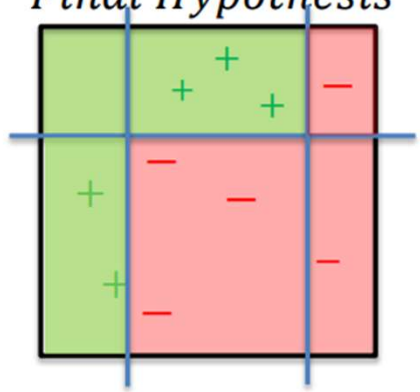
In summary, that's what we computed along the three rounds



Adaboost – Final Hypothesis

$H(x) = sign$ [.42  +.65  +.92 ]

Final Hypothesis



Reweighting vs Resampling

- Example weights might be harder to deal with
  - Some learning methods can’t use weights on examples
- We can resample instead:
  - Draw a bootstrap sample from the data with the probability of drawing each example proportional to its weight
- Reweighting usually works better but resampling is easier to implement

Summary: Boosting vs. Bagging

- Bagging doesn’t work so well with stable models. Boosting might still help.
- Boosting might hurt performance on noisy datasets. Bagging doesn’t have this problem.
- On average, boosting helps more than bagging, but it is also more common for boosting to hurt performance.
- Bagging is easier to parallelize.

Readings

- **Pattern Recognition and Machine Learning by Christopher M. Bishop – Chapter 14**
- **Explaining AdaBoost by Robert E. Schapire**