Introduction to AdaBoost

Nikolaos Nikolaou

School of Computer Science
University of Manchester

MANCHESTER
1824
The University of Manchester

What is AdaBoost?

- Bagging = “Parallel”
- AdaBoost = “Sequential”

- Originally **binary classification**

- Extensions for multiclass, regression, ranking, …

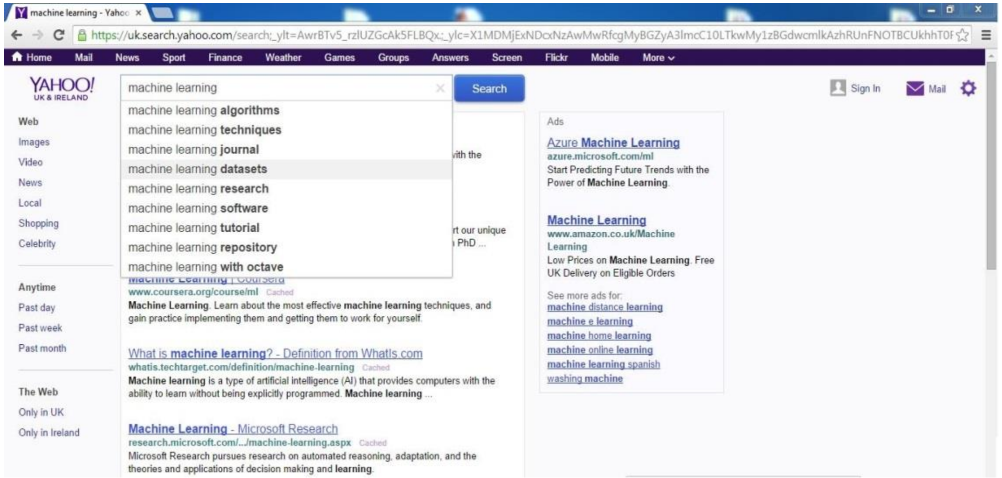
Ever Used AdaBoost?

Ever Used AdaBoost?

Ever played Call of Duty?

Ever Used AdaBoost?

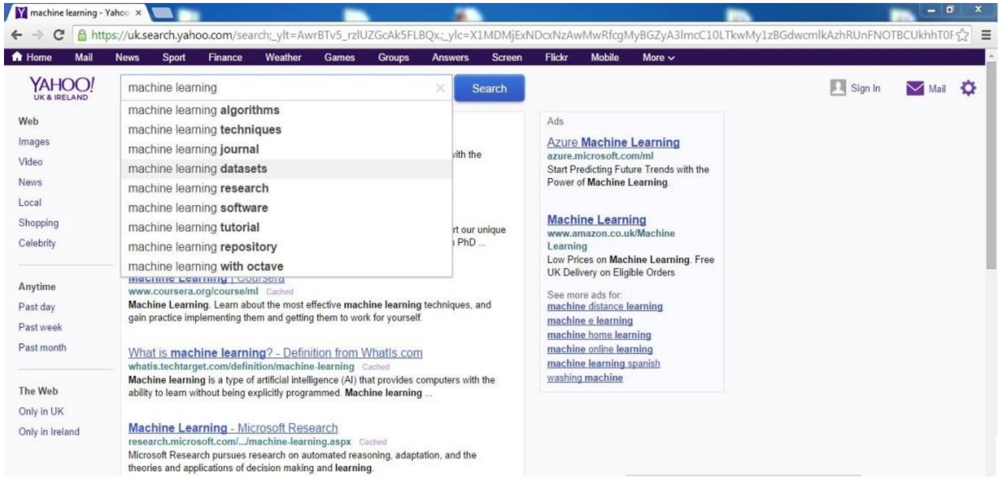
Ever played Call of Duty?



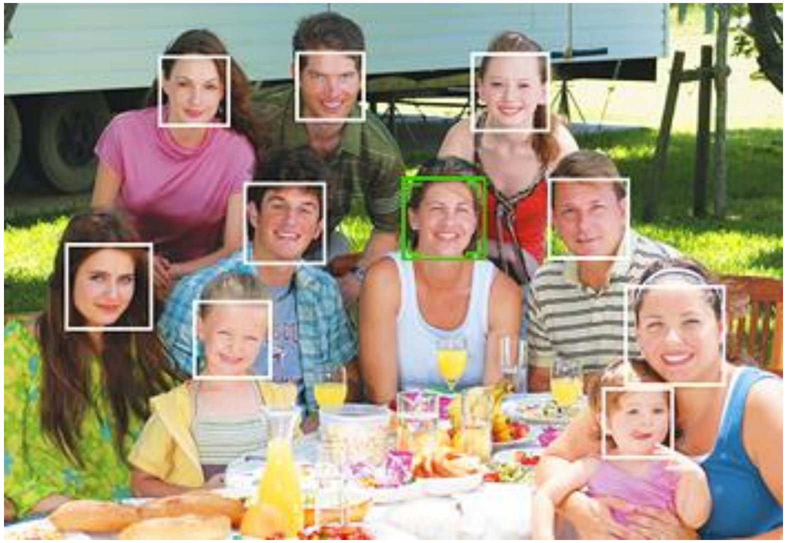
Or used the Yahoo search engine…?

Ever Used AdaBoost?

Ever played Call of Duty?



Or used the Yahoo search engine…?



Or have a phone with a camera…?

History of the Concept (1)

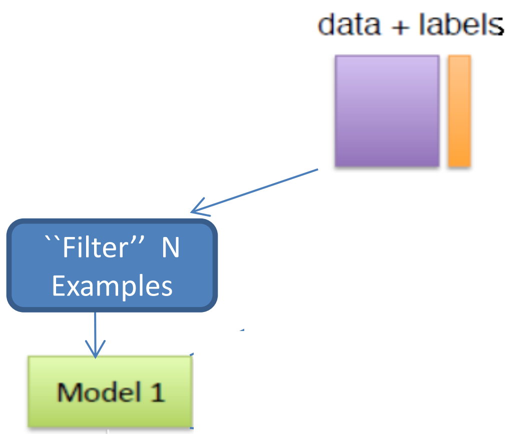
- Can we turn a weak learner into a strong learner? (Kearns, 1988)

History of the Concept (1)

- Can we turn a weak learner into a strong learner? (Kearns, 1988)
- **Yes**! (Schapire, 1990)

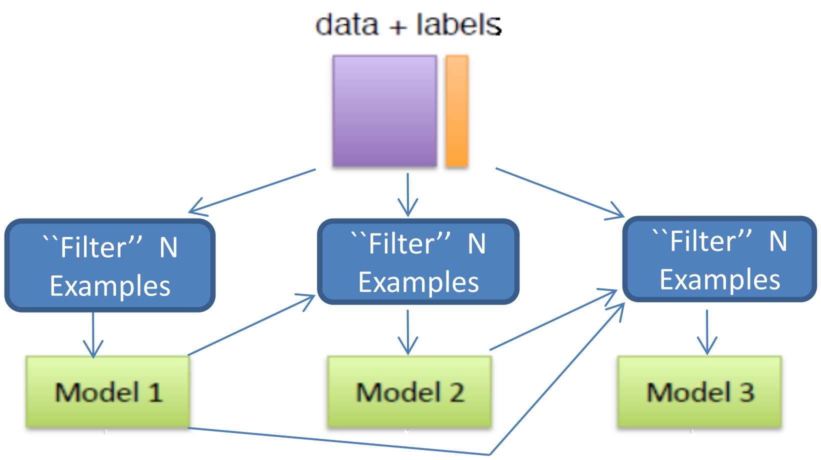
History of the Concept (1)

- Can we turn a weak learner into a strong learner? (Kearns, 1988)
- **Yes**! (Schapire, 1990)



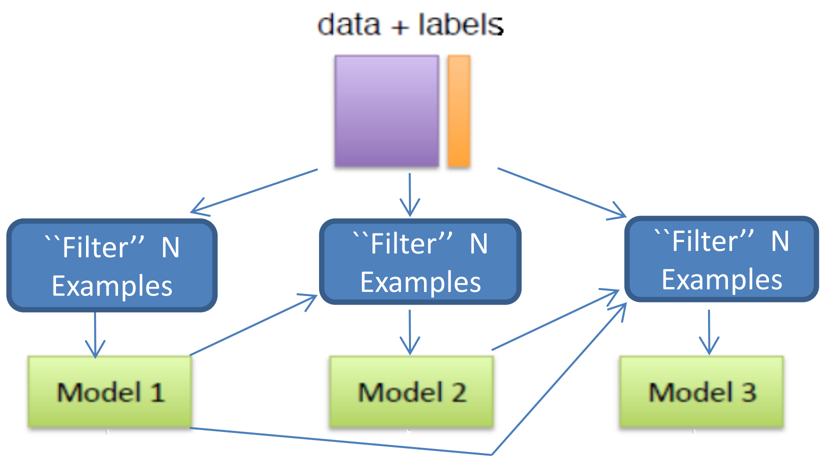
History of the Concept (1)

- Can we turn a weak learner into a strong learner? (Kearns, 1988)
- **Yes**! (Schapire, 1990)



History of the Concept (1)

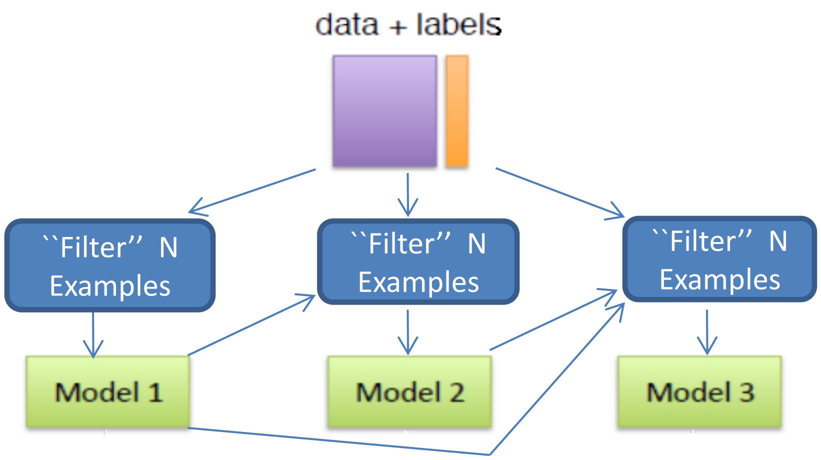
- Can we turn a weak learner into a strong learner? (Kearns, 1988)
- **Yes**! (Schapire, 1990)



Majority Vote better than Model 1

History of the Concept (1)

- Can we turn a weak learner into a strong learner? (Kearns, 1988)
- **Yes**! (Schapire, 1990)



Majority Vote better than Model 1

- Adaptive boosting (**Adaboost**): M models built "adaptively" (Freund & Schapire, 1997)

2003 Gödel Prize

History of the Concept (2)

- Cofidence-rated predictions instead of majority vote (Schapire & Singer, 1998)
- Since then lots of variants and applications…

History of the Concept (2)

- Cofidence-rated predictions instead of majority vote (Schapire & Singer, 1998)
- Since then lots of variants and applications…

Top 10 algorithms in data mining

Xindong Wu · Vipin Kumar · J. Ross Quinlan · Joydeep Ghosh · Qiang Yang · Hiroshi Motoda · Geoffrey J. McLachlan · Angus Ng · Bing Liu · Philip S. Yu · Zhi-Hua Zhou · Michael Steinbach · David J. Hand · Dan Steinberg

Received: 9 July 2007 / Revised: 28 September 2007 / Accepted: 8 October 2007
Published online: 4 December 2007

Abstract This paper presents the top 10 data mining algorithms identified by the IEEE International Conference on Data Mining (ICDM) in December 2006: C4.5, *k*-Means, SVM, Apriori, EM, PageRank, AdaBoost, *k*NN, Naive Bayes, and CART. These top 10 algorithms are among the most influential data mining algorithms in the research community. With each algorithm, we provide a description of the algorithm, discuss the impact of the algorithm, and review current and further research on the algorithm. These 10 algorithms cover classification,

History of the Concept (2)

- Cofidence-rated predictions instead of majority vote (Schapire & Singer, 1998)
- Since then lots of variants and applications…

Top 10 algorithms in data mining

Xindong Wu · Vipin Kumar · J. Ross Quinlan · Joydeep Ghosh · Qiang Yang · Hiroshi Motoda · Geoffrey J. McLachlan · Angus Ng · Bing Liu · Philip S. Yu · Zhi-Hua Zhou · Michael Steinbach · David J. Hand · Dan Steinberg

Received: 9 July 2007 / Revised: 28 September 2007 / Accepted: 8 October 2007
Published online: 4 December 2007

Abstract This paper presents the top 10 data mining algorithms identified by the IEEE International Conference on Data Mining (ICDM) in December 2006: C4.5, *k*-Means, SVM, Apriori, EM, PageRank, AdaBoost, *k*NN, Naive Bayes, and CART. These top 10 algorithms are among the most influential data mining algorithms in the research community. With each algorithm, we provide a description of the algorithm, discuss the impact of the algorithm, and review current and further research on the algorithm. These 10 algorithms cover classification,

An Empirical Comparison of Supervised Learning Algorithms

Rich Caruana CARUANA@CS.CORNELL.EDU
Alexandru Niculescu-Mizil ALEXN@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

With excellent performance on all eight metrics, calibrated boosted trees were the best learning algorithm overall. Random forests are close second, followed by uncalibrated bagged trees, calibrated SVMs, and uncalibrated neural nets. The models that performed

History of the Concept (2)

- Cofidence-rated predictions instead of majority vote (Schapire & Singer, 1998)
- Since then lots of variants and applications…

Top 10 algorithms in data mining

Xindong Wu · Vipin Kumar · J. Ross Quinlan · Joydeep Ghosh · Qiang Yang · Hiroshi Motoda · Geoffrey J. McLachlan · Angus Ng · Bing Liu · Philip S. Yu · Zhi-Hua Zhou · Michael Steinbach · David J. Hand · Dan Steinberg

Received: 9 July 2007 / Revised: 28 September 2007 / Accepted: 8 October 2007
Published online: 4 December 2007

Abstract This paper presents the top 10 data mining algorithms identified by the IEEE International Conference on Data Mining (ICDM) in December 2006: C4.5, *k*-Means, SVM, Apriori, EM, PageRank, AdaBoost, *k*NN, Naive Bayes, and CART. These top 10 algorithms are among the most influential data mining algorithms in the research community. With each algorithm, we provide a description of the algorithm, discuss the impact of the algorithm, and review current and further research on the algorithm. These 10 algorithms cover classification,

An Empirical Comparison of Supervised Learning Algorithms

Rich Caruana CARUANA@CS.CORNELL.EDU
Alexandru Niculescu-Mizil ALEXN@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

With excellent performance on all eight metrics, calibrated boosted trees were the best learning algorithm overall. Random forests are close second, followed by uncalibrated bagged trees, calibrated SVMs, and uncalibrated neural nets. The models that performed

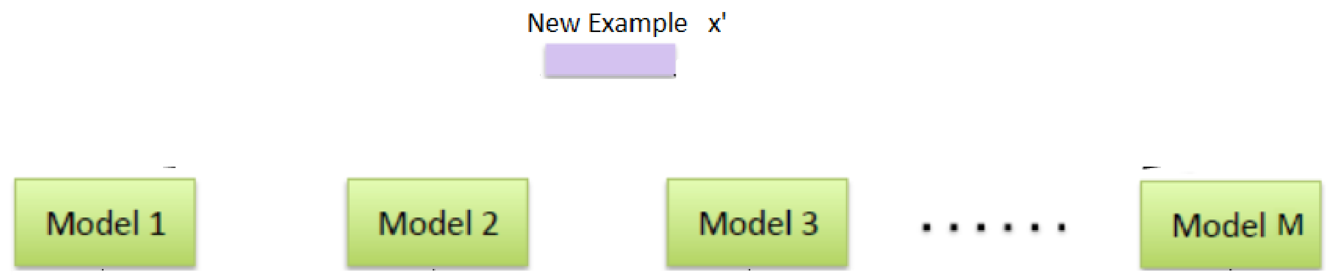
kaggle™

AdaBoost as an Ensemble Method

- Prediction: **Weighted majority vote** among M weak learners

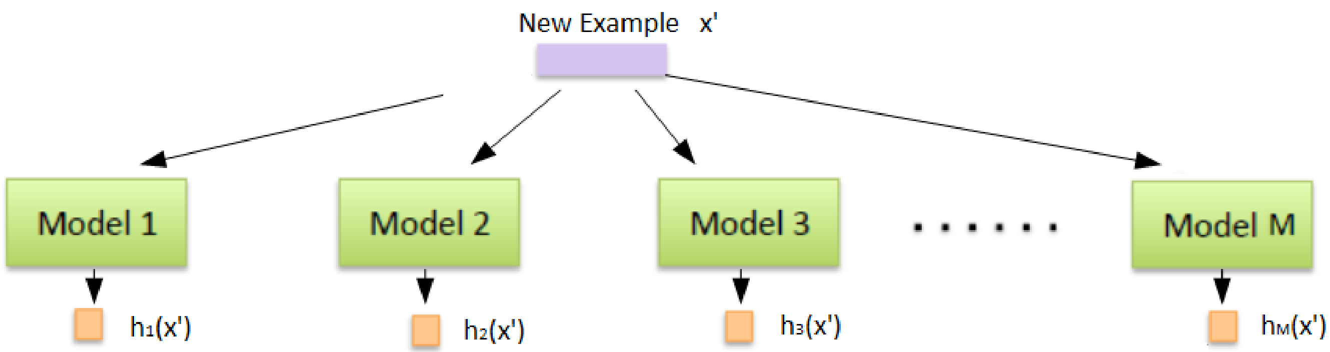
AdaBoost as an Ensemble Method

- Prediction: **Weighted majority vote** among M weak learners



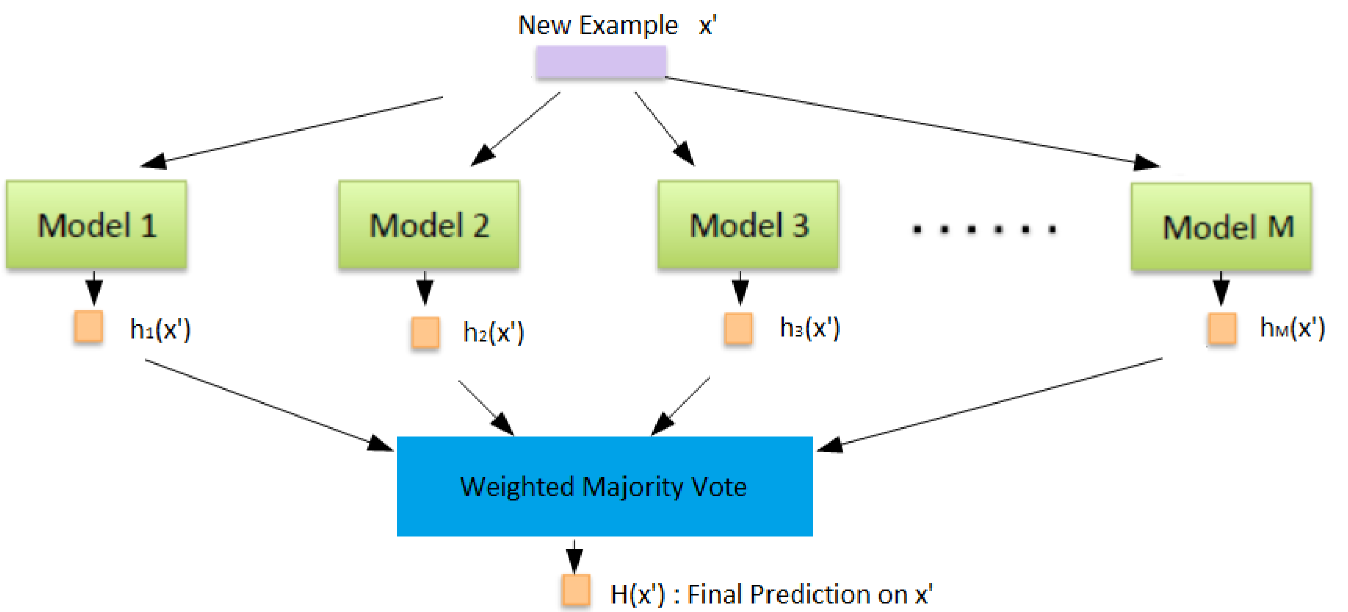
AdaBoost as an Ensemble Method

- Prediction: **Weighted majority vote** among M weak learners



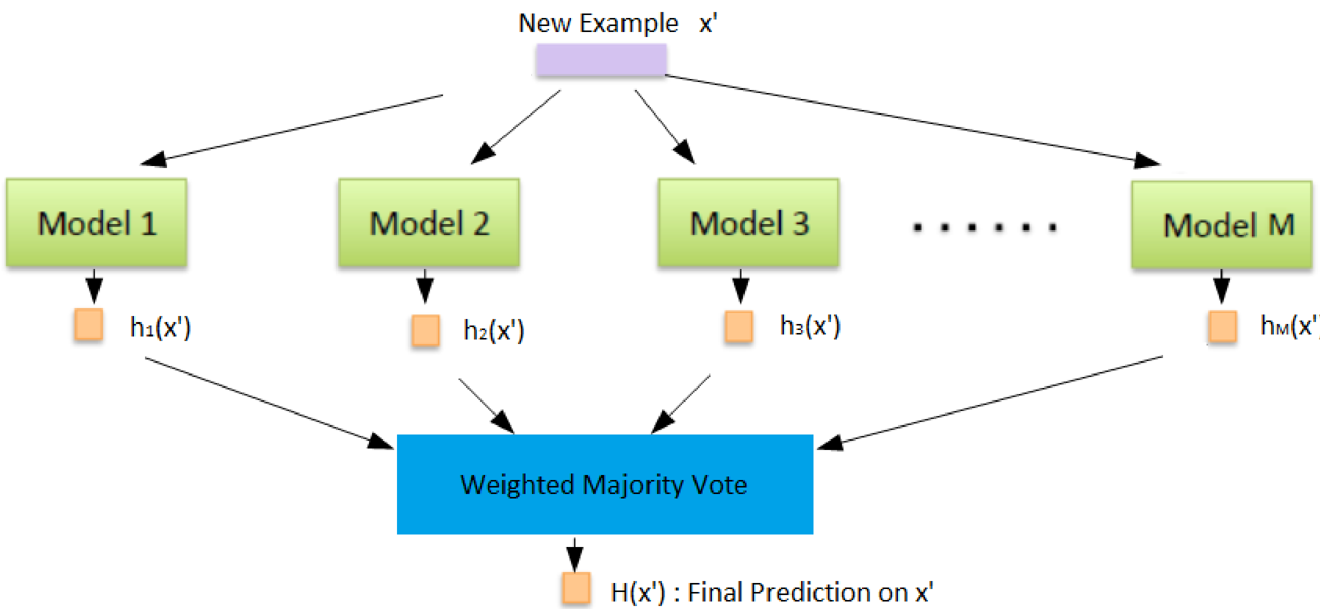
AdaBoost as an Ensemble Method

- Prediction: **Weighted majority vote** among M weak learners



AdaBoost as an Ensemble Method

- Prediction: **Weighted majority vote** among M weak learners



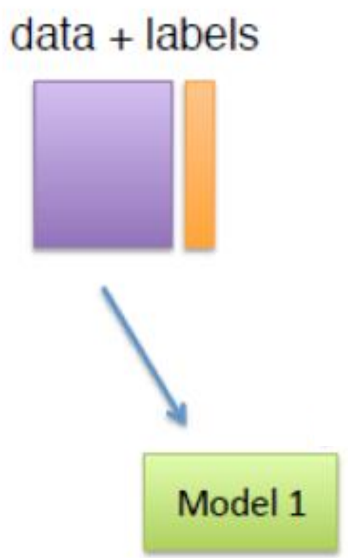
- Can apply to any supervised base learner...

How is AdaBoost "Adaptive"?

- Idea: Construct strong model **sequentially** by combining multiple weak models

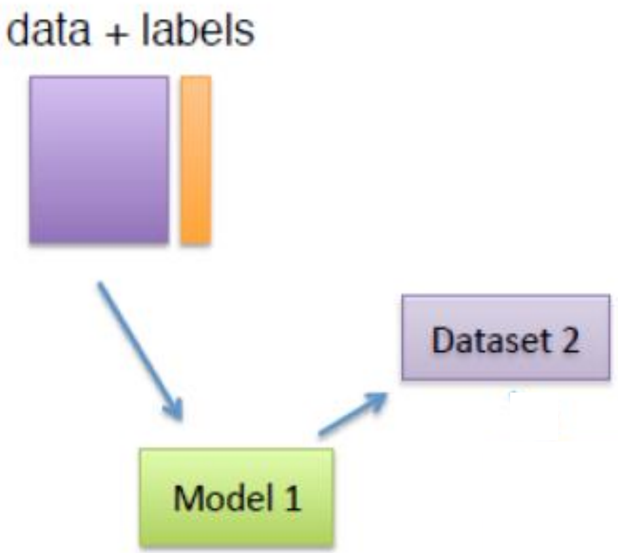
How is AdaBoost "Adaptive"?

- Idea: Construct strong model **sequentially** by combining multiple weak models



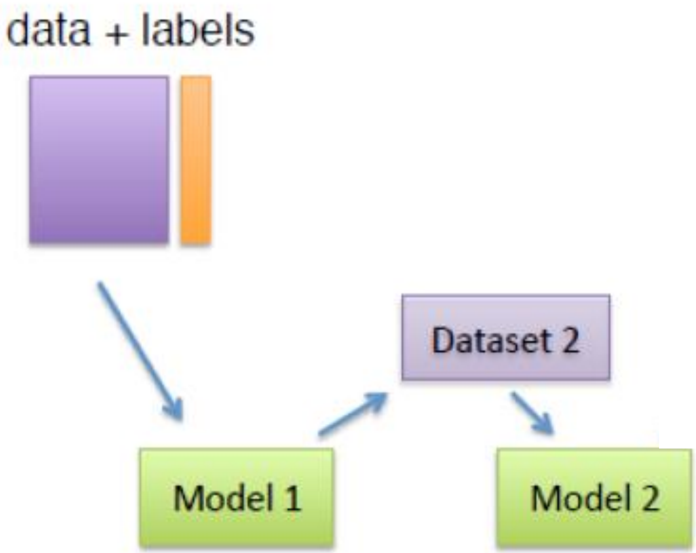
How is AdaBoost "Adaptive"?

- Idea: Construct strong model **sequentially** by combining multiple weak models



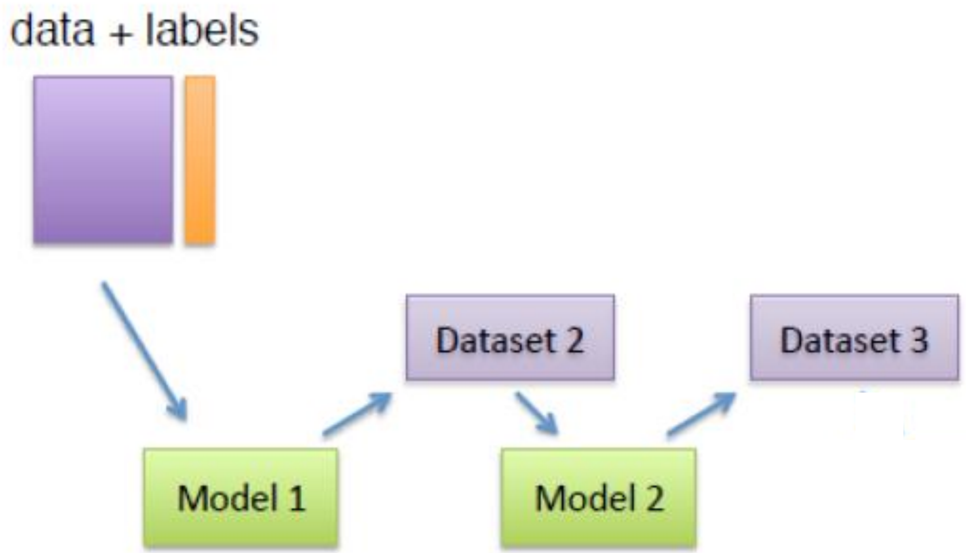
How is AdaBoost "Adaptive"?

- Idea: Construct strong model **sequentially** by combining multiple weak models



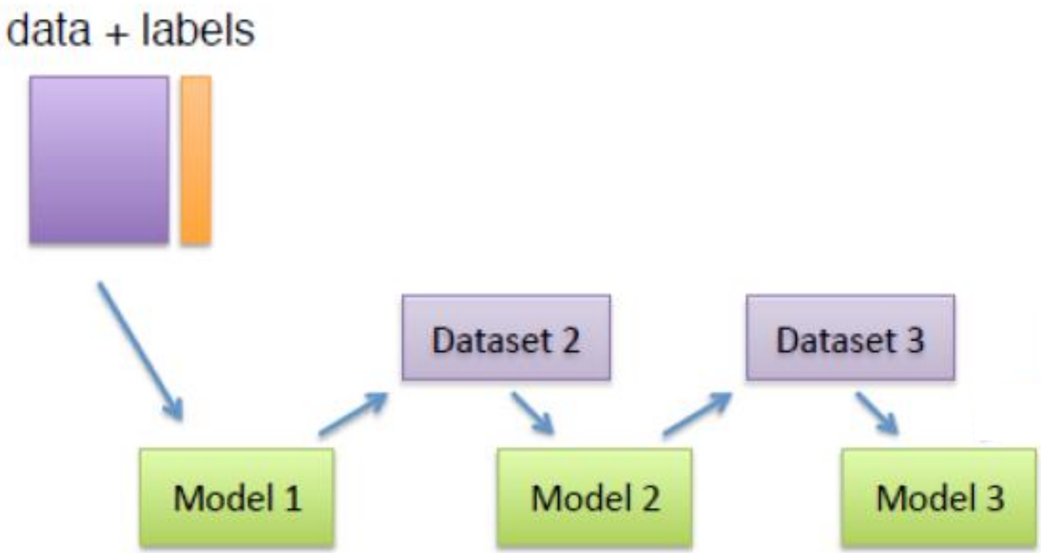
How is AdaBoost "Adaptive"?

- Idea: Construct strong model **sequentially** by combining multiple weak models



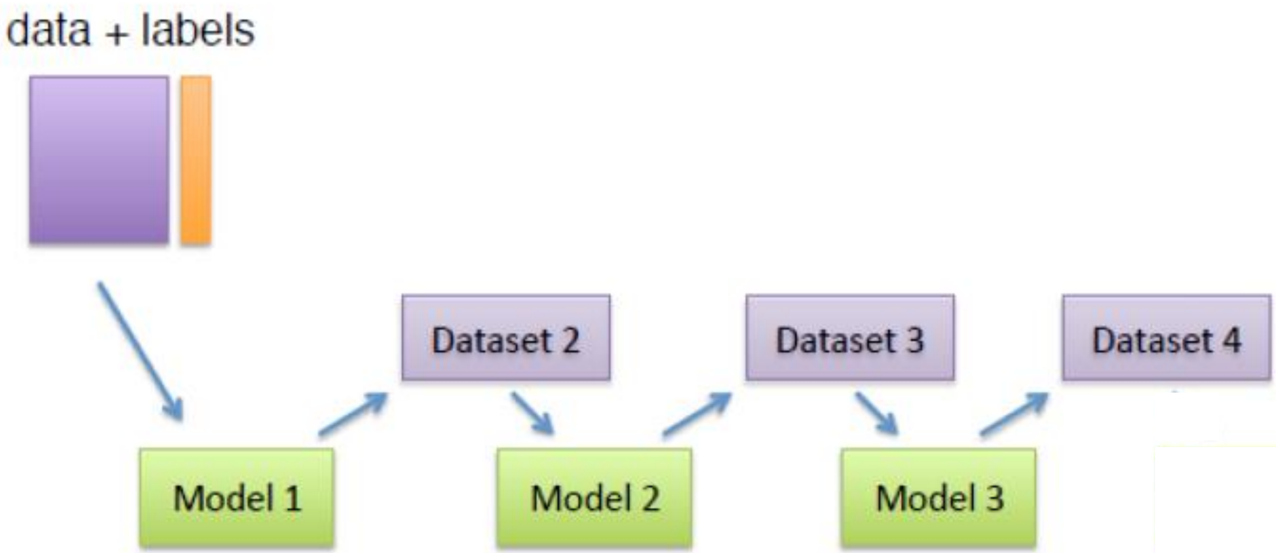
How is AdaBoost “Adaptive”?

- Idea: Construct strong model **sequentially** by combining multiple weak models



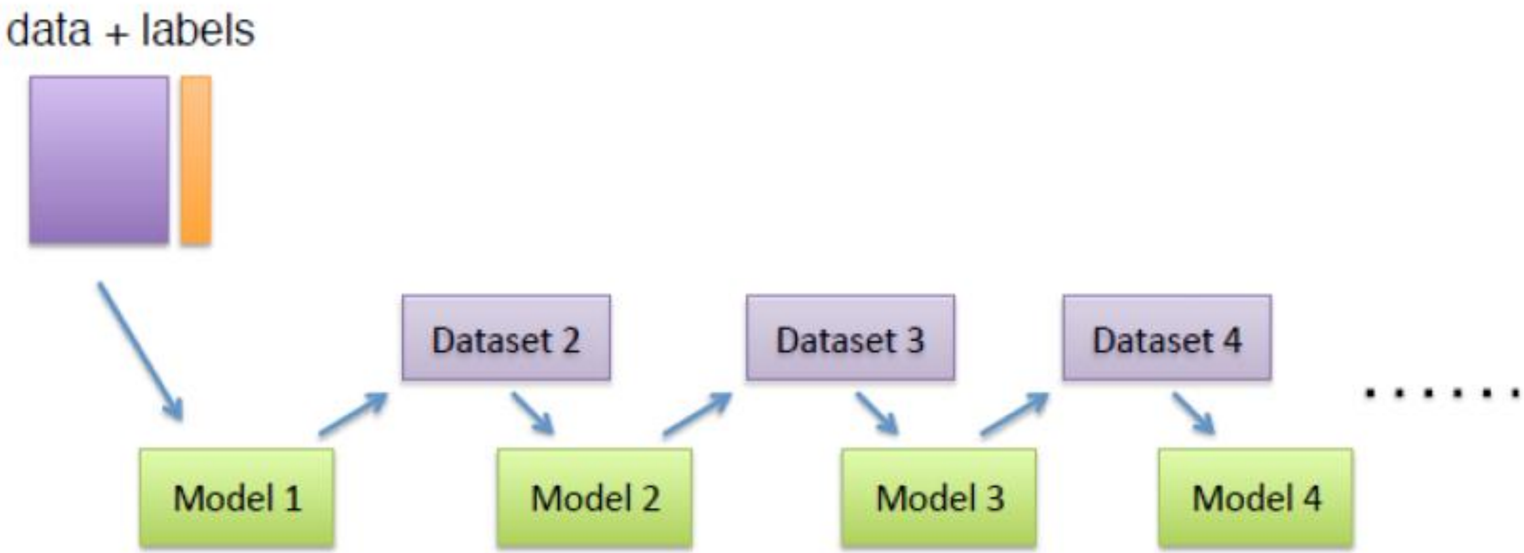
How is AdaBoost "Adaptive"?

- Idea: Construct strong model **sequentially** by combining multiple weak models



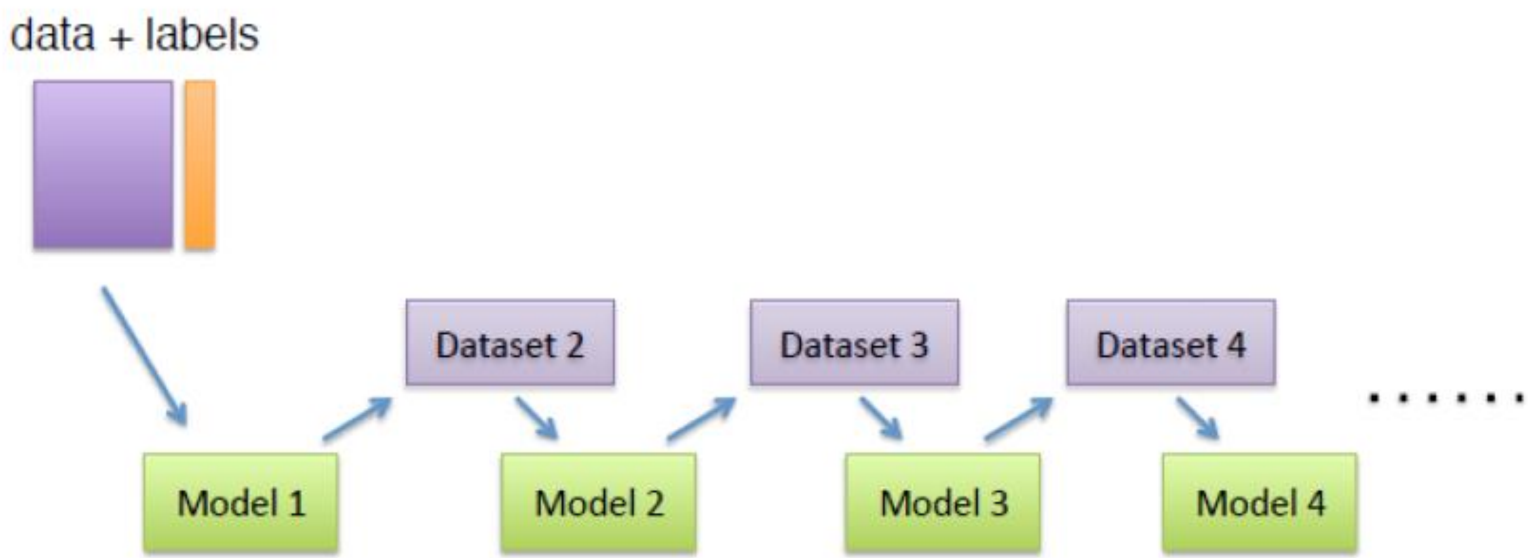
How is AdaBoost "Adaptive"?

- Idea: Construct strong model **sequentially** by combining multiple weak models



How is AdaBoost "Adaptive"?

- Idea: Construct strong model **sequentially** by combining multiple weak models

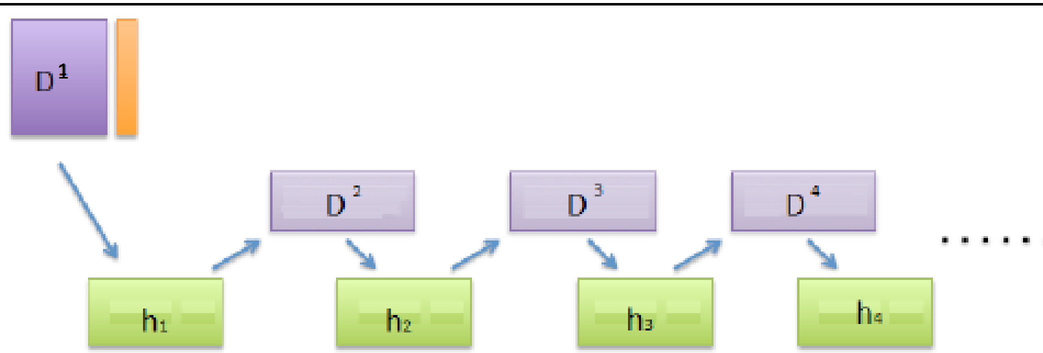


- Each model tries to **correct the mistakes of the previous one**

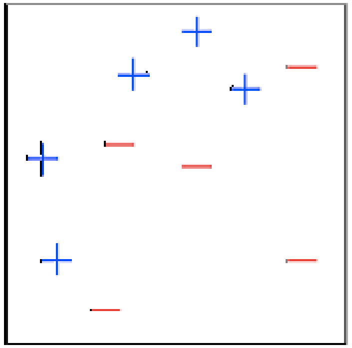
AdaBoost: Algorithm Outline

Algorithm 1: AdaBoost Sketch

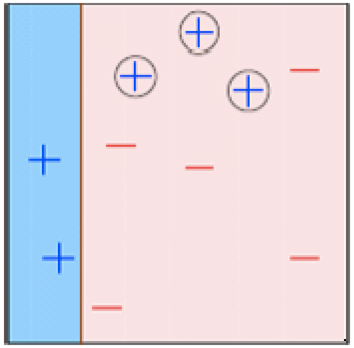
Input: Training Data $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, Number of rounds M.
Training:
Define a weight distribution over the examples $D_i^1 = \frac{1}{N}$, for $i = 1, 2, \ldots, N$.
for round $j = 1$ to M **do**
  Build a model h_j from the training set using distribution D^j.
  Update D^{j+1} from D^j:
    Increase weights of examples misclasified by h_j.
    Decrease weights of examples correctly classified by h_j.
end for
Prediction: For a new example x', output the weighted (confidence-rated) majority vote of the models $\{h_1, h_2, \ldots, h_M\}$.



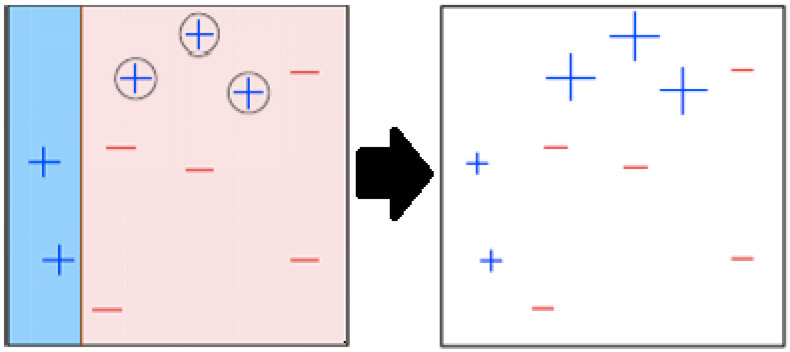
AdaBoost Example



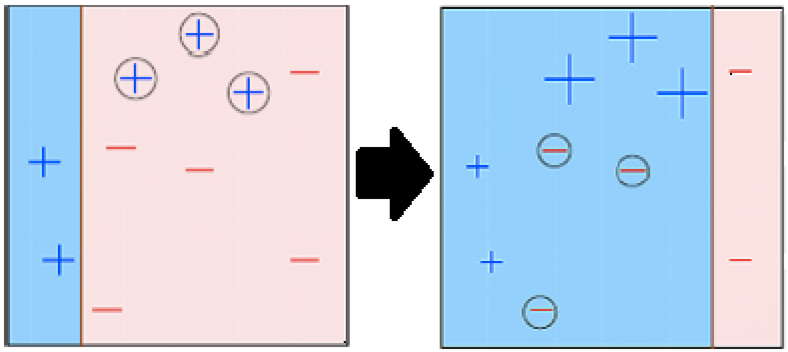
AdaBoost Example



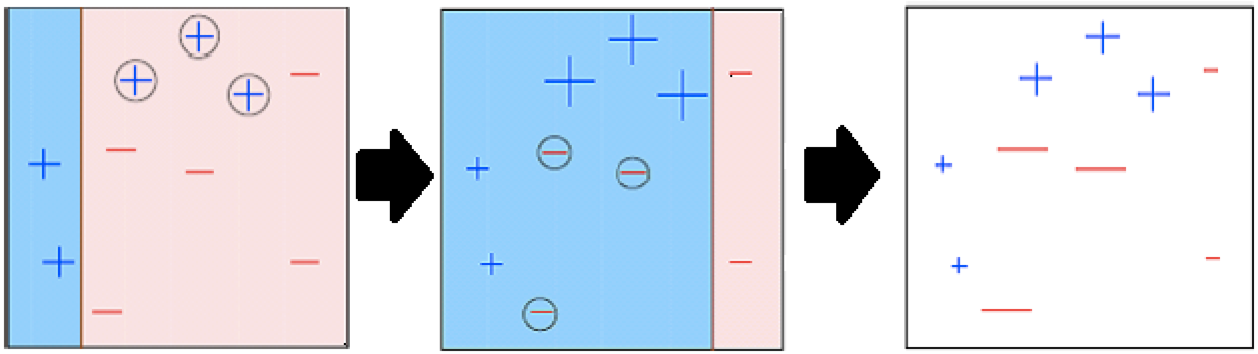
AdaBoost Example



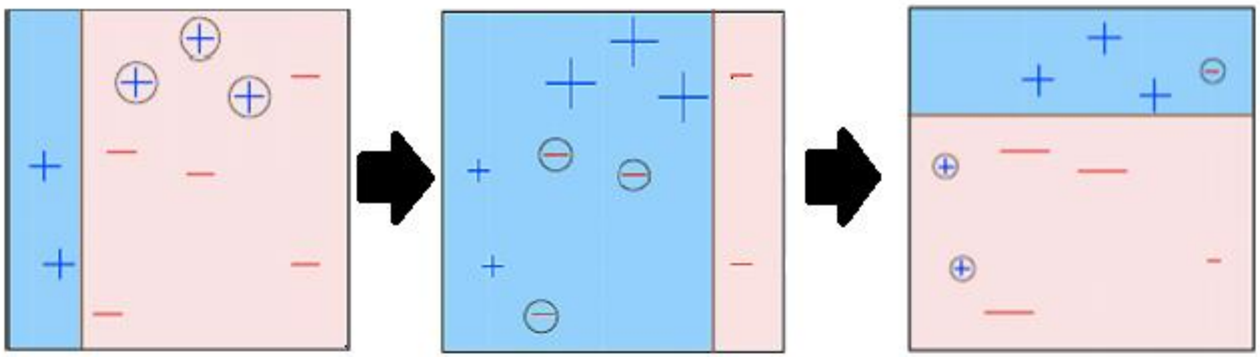
AdaBoost Example



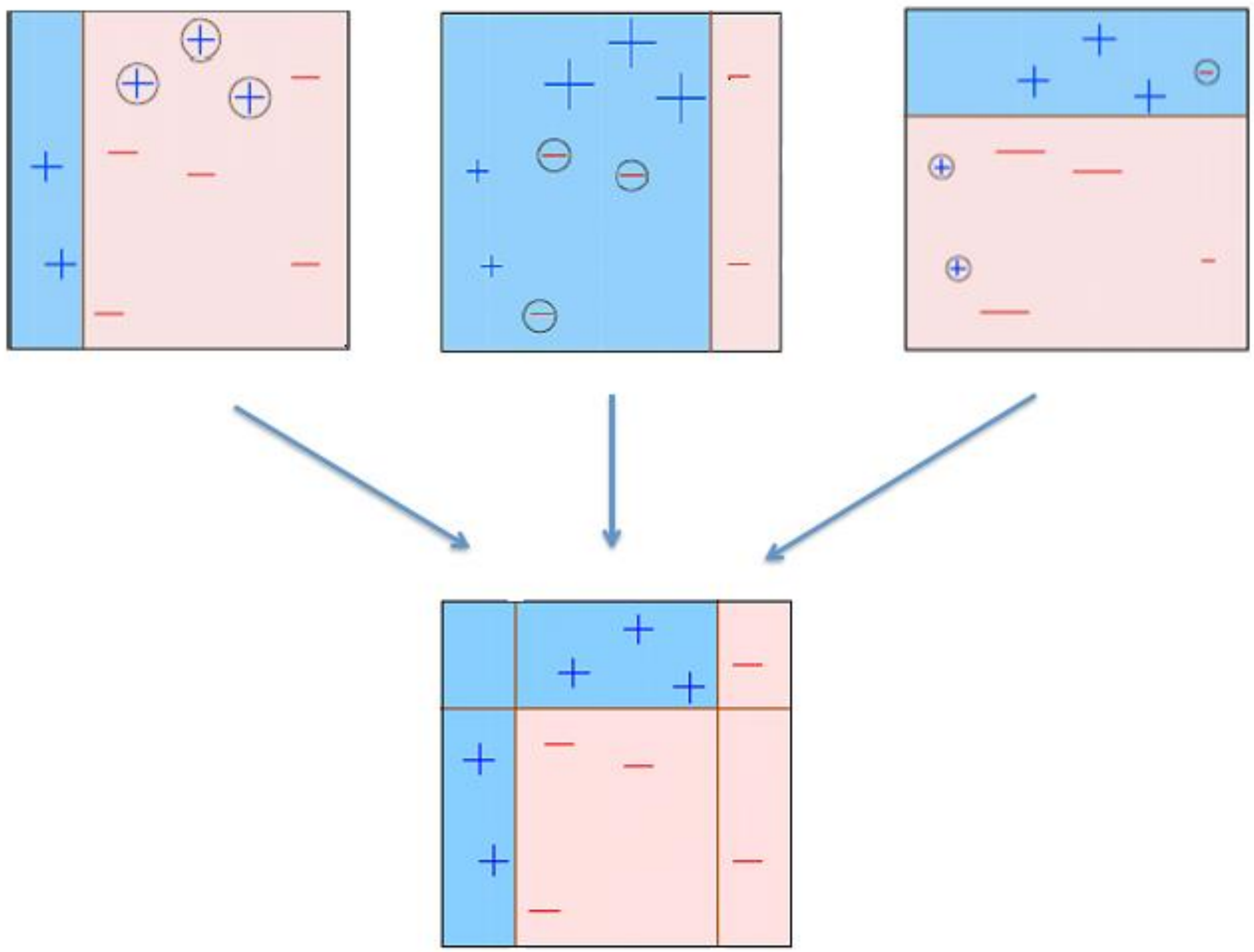
AdaBoost Example



AdaBoost Example



AdaBoost Example



AdaBoost vs Bagging

- Both train many models on different versions of initial dataset and then aggregate, but…

AdaBoost vs Bagging

- Both train many models on different versions of initial dataset and then aggregate, but…

BAGGING	**ADABOOST**
Resample dataset	Resample or reweight dataset

AdaBoost vs Bagging

- Both train many models on different versions of initial dataset and then aggregate, but…

BAGGING	ADABOOST
Resample dataset	Resample or reweight dataset
Builds base models in parallel	Builds base models sequentially

AdaBoost vs Bagging

- Both train many models on different versions of initial dataset and then aggregate, but…

BAGGING	ADABOOST
Resample dataset	Resample or reweight dataset
Builds base models in parallel	Builds base models sequentially
Reduces variance (doesn’t work well with e.g. decision stumps)	Also reduces bias (works well with stumps)

AdaBoost vs Bagging

- Both train many models on different versions of initial dataset and then aggregate, but…

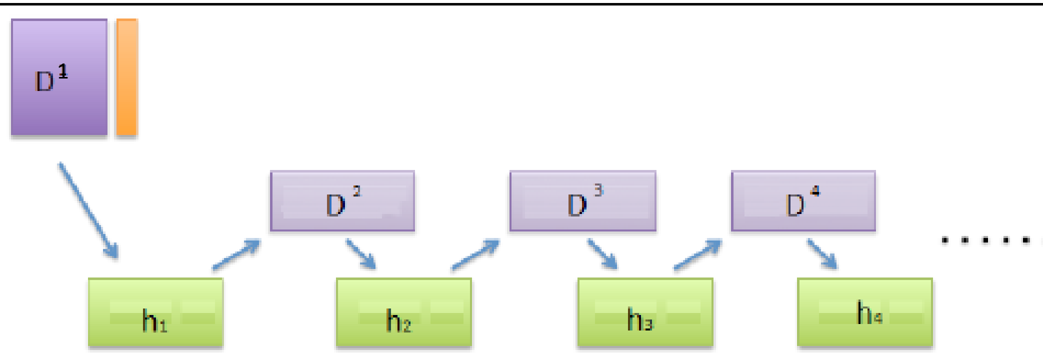
BAGGING	ADABOOST
Resample dataset	Resample or reweight dataset
Builds base models in parallel	Builds base models sequentially
Reduces variance (doesn't work well with e.g. decision stumps)	Also reduces bias (works well with stumps)

- So, bagging & AdaBoost **fundamentally different**!

AdaBoost: Algorithm outline

Algorithm 1: AdaBoost Sketch

Input: Training Data $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, Number of rounds M.
Training:
Define a weight distribution over the examples $D_i^1 = \frac{1}{N}$, for $i = 1, 2, \ldots, N$.
for round $j = 1$ to M **do**
  Build a model h_j from the training set using distribution D^j.
  Update D^{j+1} from D^j:
    Increase weights of examples misclasified by h_j.
    Decrease weights of examples correctly classified by h_j.
end for
Prediction: For a new example $\mathbf{x}'$, output the weighted (confidence-rated) majority vote of the models $\{h_1, h_2, \ldots, h_M\}$.



AdaBoost: Algorithm

Algorithm 2 AdaBoost

Input: Training Data $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of rounds M.
Training:
$D_i^1 = \frac{1}{N}$, for $i = 1, 2, \ldots, N$.
for $j = 1$ to M **do**

Define $\epsilon_j = \sum_{i:h_j(x_i) \neq y_i} D_i^j$.

Obtain a hypothesis h_j that minimizes ϵ_j and satisfies the condition $\epsilon_j < \frac{1}{2}$.

$\alpha_j = \frac{1}{2} \log\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$.

$D_i^{j+1} = e^{-y_i h_j(x_i) \alpha_j} D_i^j$.

$D_i^{j+1} = \frac{D_i^{j+1}}{\sum_{i=1}^N D_i^{j+1}}$.

end for
Prediction: $H(\boldsymbol{x}') = sign\left[\sum_{j=1}^M \alpha_j h_j(\boldsymbol{x}')\right]$.

$y \in \{-1, 1\}$

AdaBoost: Algorithm

Algorithm 2 AdaBoost

Input: Training Data $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of rounds M.
Training:
$D_i^1 = \frac{1}{N}$, for $i = 1, 2, \ldots, N$.
for $j = 1$ to M **do**

Define $\boxed{\epsilon_j = \sum_{i:h_j(x_i) \neq y_i} D_i^j.}$

εj: weighted error of the j-th model

Obtain a hypothesis h_j that minimizes ϵ_j and satisfies the condition $\epsilon_j < \frac{1}{2}$.

$\alpha_j = \frac{1}{2} \log\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$.

$D_i^{j+1} = e^{-y_i h_j(x_i) \alpha_j} D_i^j$.

$D_i^{j+1} = \frac{D_i^{j+1}}{\sum_{i=1}^N D_i^{j+1}}$.

end for
Prediction: $H(\boldsymbol{x}') = sign\left[\sum_{j=1}^M \alpha_j h_j(\boldsymbol{x}')\right]$.

$y \in \{-1, 1\}$

AdaBoost: Algorithm

Algorithm 2 AdaBoost

Input: Training Data $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of rounds M.
Training:
$D_i^1 = \frac{1}{N}$, for $i = 1, 2, \ldots, N$.
for $j = 1$ to M **do**
  Define $\boxed{\epsilon_j = \sum_{i:h_j(x_i) \neq y_i} D_i^j.}$

εj: weighted error of the j-th model

  Obtain a hypothesis h_j that minimizes ϵ_j and satisfies the condition $\epsilon_j < \frac{1}{2}$.
  $\boxed{\alpha_j = \frac{1}{2} \log\left(\frac{1-\epsilon_j}{\epsilon_j}\right).}$

αj: ``confidence'' of the j-th model

  $D_i^{j+1} = e^{-y_i h_j(x_i)\alpha_j} D_i^j.$
  $D_i^{j+1} = \frac{D_i^{j+1}}{\sum_{i=1}^N D_i^{j+1}}.$
end for
Prediction: $H(\boldsymbol{x}') = sign\left[\sum_{j=1}^M \alpha_j h_j(\boldsymbol{x}')\right].$

$$y \in \{-1, 1\}$$

AdaBoost: Algorithm

Algorithm 2 AdaBoost

Input: Training Data $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of rounds M.
Training:
$D_i^1 = \frac{1}{N}$, for $i = 1, 2, \ldots, N$.
for $j = 1$ to M **do**

Define $\boxed{\epsilon_j = \sum_{i:h_j(x_i)\neq y_i} D_i^j.}$

ε

Obtain a hypothesis h_j that minimizes ϵ_j and satisfies the condition $\epsilon_j < \frac{1}{2}$.

$\boxed{\alpha_j = \frac{1}{2}\log\left(\frac{1-\epsilon_j}{\epsilon_j}\right).}$

$D_i^{j+1} = e^{-y_i h_j(x_i)\alpha_j} D_i^j.$

$D_i^{j+1} = \frac{D_i^{j+1}}{\sum_{i=1}^N D_i^{j+1}}.$

end for
Prediction: $H(\boldsymbol{x}') = sign\left[\sum_{j=1}^M \alpha_j h_j(\boldsymbol{x}')\right].$

ϵ_j: weighted error of the j-th model

α_j: ``confidence'' of the j-th model

Update weights for next iteration

After normalization we have: $\sum_{i=1}^{N} D_i^{j+1} = 1$

$y \in \{-1, 1\}$

AdaBoost: Prediction

- A new example **x**' arrives
- Each weak learner's vote $h_j(\mathbf{x}') \in \{-1,1\}$ is weighted by its confidence α_j to get the final prediction

$$H(\boldsymbol{x}') = sign\left[\sum_{j=1}^{M} \alpha_j h_j(\boldsymbol{x}')\right]$$

Adaboost: Training (1)

- In each training round:

  Define $\epsilon_j = \sum_{i:h_j(\mathbf{x}_i)\neq y_i} D_i^j$.

  Obtain a hypothesis h_j that minimizes $\epsilon_j < \frac{1}{2}$

  $\alpha_j = \frac{1}{2}\log\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$.

  $D_i^{j+1} = e^{-y_i h_j(\mathbf{x}_i)\alpha_j} D_i^j$.

Adaboost: Training (1)

- In each training round:

  Define $\epsilon_j = \sum_{i:h_j(\mathbf{x}_i) \neq y_i} D_i^j$.

  Obtain a hypothesis h_j that minimizes $\epsilon_j < \frac{1}{2}$

  $\alpha_j = \frac{1}{2} \log\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$.

  $D_i^{j+1} = e^{-y_i h_j(\mathbf{x}_i)\alpha_j} D_i^j$.

QUESTION:

What if $\epsilon_j > \frac{1}{2}$?

Adaboost: Training (1)

- In each training round:

  Define $\epsilon_j = \sum_{i:h_j(\mathbf{x}_i)\neq y_i} D_i^j$.

  Obtain a hypothesis h_j that minimizes $\epsilon_j < \frac{1}{2}$

  $\alpha_j = \frac{1}{2}\log\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$.

  $D_i^{j+1} = e^{-y_i h_j(\mathbf{x}_i)\alpha_j} D_i^j$.

QUESTION:

What if $\epsilon_j > \frac{1}{2}$?

ANSWER: Just **flip** predictions!

Resulting learner **has** $\epsilon_j < \frac{1}{2}$

Adaboost: Training (2)

- In each training round:

  Define $\epsilon_j = \sum_{i:h_j(\mathbf{x}_i)\neq y_i} D_i^j$.

  Obtain a hypothesis h_j that minimizes ϵ_j

  $\alpha_j = \frac{1}{2}\log\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$.

  $D_i^{j+1} = e^{-y_i h_j(\mathbf{x}_i)\alpha_j} D_i^j$.

Adaboost: Training (2)

- In each training round:

  Define $\epsilon_j = \sum_{i:h_j(\mathbf{x}_i) \neq y_i} D_i^j$.

  Obtain a hypothesis h_j that minimizes ϵ_j

  $\alpha_j = \frac{1}{2} \log\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$.

  But $\epsilon_j < \frac{1}{2}$, so $\alpha_j > 0$

  $D_i^{j+1} = e^{-y_i h_j(\mathbf{x}_i) \alpha_j} D_i^j$.

Adaboost: Training (2)

- In each training round:

  Define $\epsilon_j = \sum_{i:h_j(\mathbf{x}_i)\neq y_i} D_i^j$.

  Obtain a hypothesis h_j that minimizes ϵ_j

  $\alpha_j = \frac{1}{2}\log\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$.

  But $\epsilon_j < \frac{1}{2}$, so $\alpha_j > 0$

  $D_i^{j+1} = e^{-y_i h_j(\mathbf{x}_i)\alpha_j} D_i^j$.

Remember, $y \in \{-1, 1\}$ so, 2 cases:

$y_i = h_j(\mathbf{x}_i) \implies y_i h_j(\mathbf{x}_i) = 1 \implies D_i^{j+1} = e^{-\alpha_j} D_i^j \implies D_i^{j+1} < D_i^j$

(h_{j+1} **can afford to pay less attention** to i-th example)

OR

$y_i \neq h_j(\mathbf{x}_i) \implies y_i h_j(\mathbf{x}_i) = -1 \implies D_i^{j+1} = e^{\alpha_j} D_i^j \implies D_i^{j+1} > D_i^j$

(h_{j+1} must **try harder** than h_j to get i-th example right)

Adaboost: Training (3)

- In each training round:

  Define $\epsilon_j = \sum_{i:h_j(\mathbf{x}_i)\neq y_i} D_i^j$.

  Obtain a hypothesis h_j that minimizes ϵ_j

  $\alpha_j = \frac{1}{2}\log\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$.

  $D_i^{j+1} = e^{-y_i h_j(\mathbf{x}_i)\alpha_j} D_i^j$.

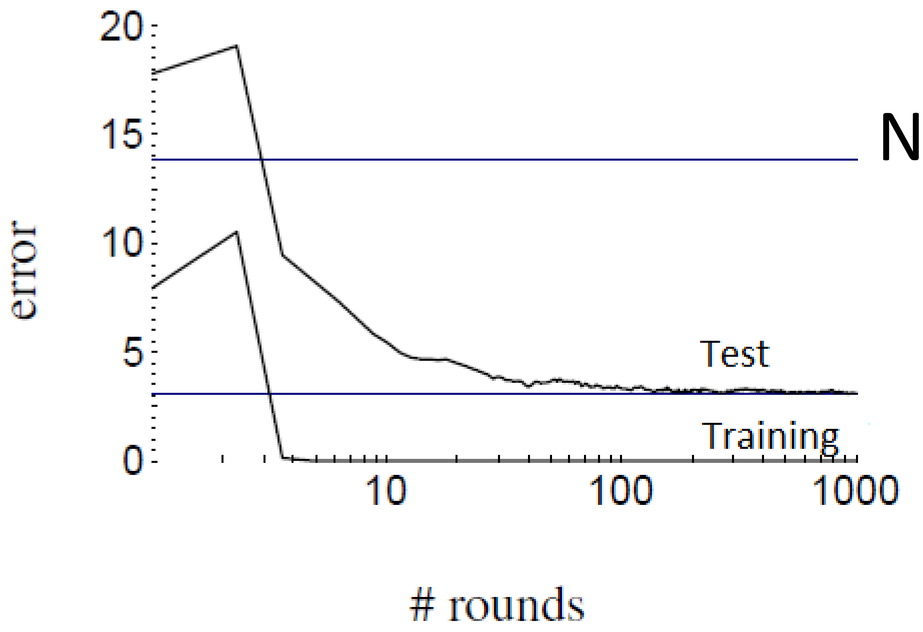
Note: weight updates are **confidence-rated**; the higher the confidence α_j , the greater the weight increase/decrease of each D_i^{j+1}

Why this α_j?

- Why the magic choice of $\alpha_j = \frac{1}{2}\log\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$?
- Beyond scope of lecture
- A consequence: 50% of new weight mass D^{j+1} assigned to examples misclassified by previous learner h_j.
  - Decorrelates consecutive learners

Resistance to Overfitting

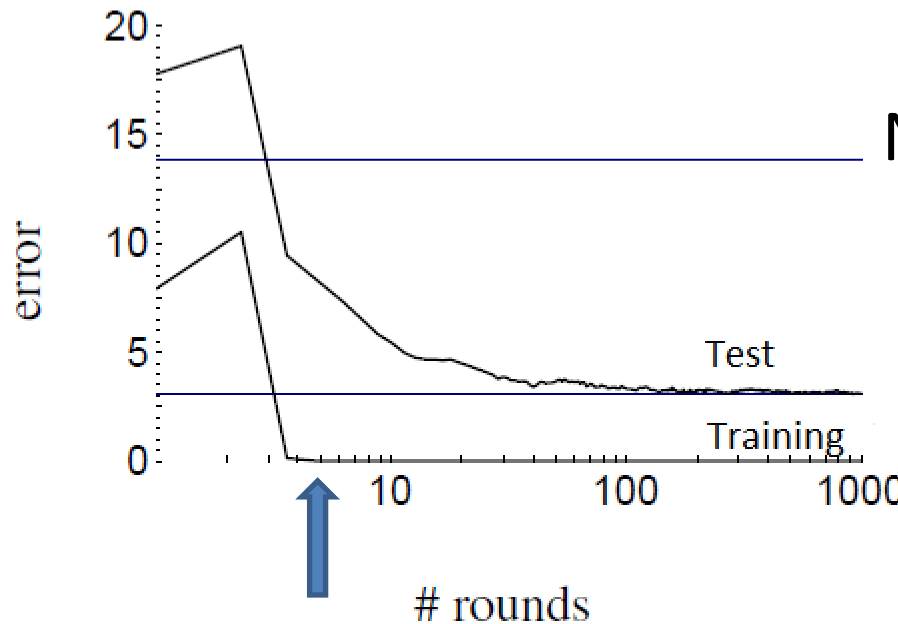
- Often we observe AdaBoost behaving like this:



Notice something strange?

Resistance to Overfitting

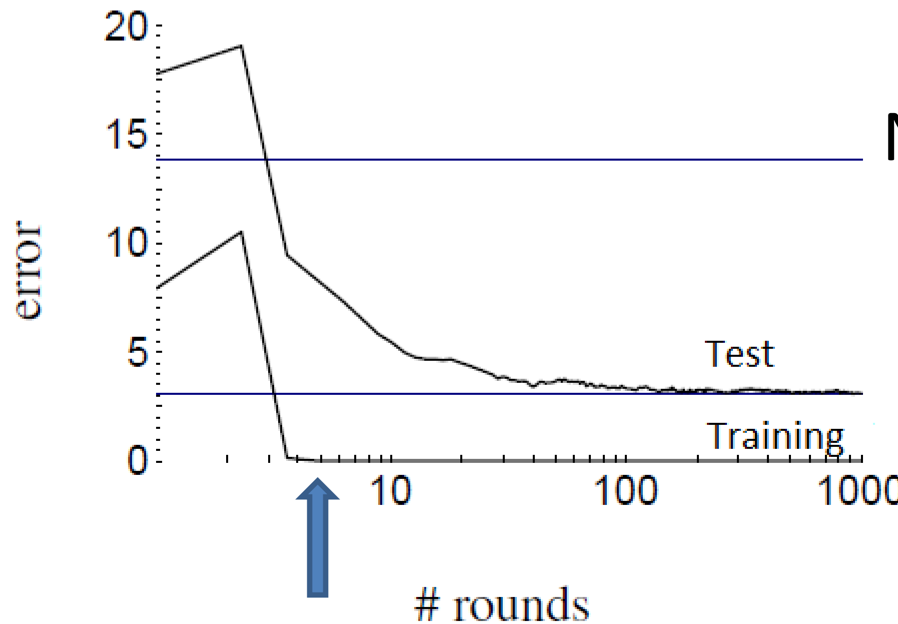
- Often we observe AdaBoost behaving like this:



Notice something strange?

Resistance to Overfitting

- Often we observe AdaBoost behaving like this:

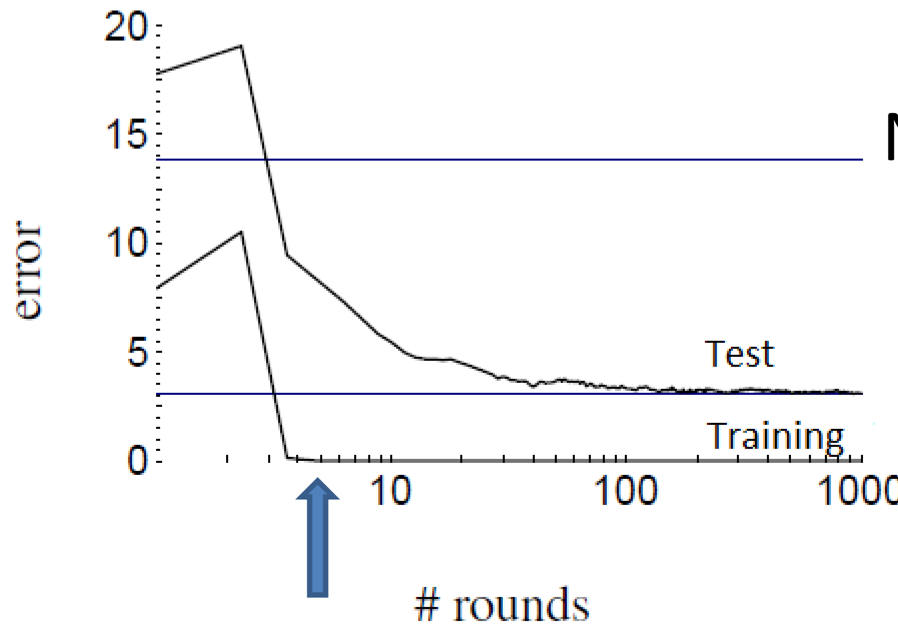


Notice something strange?

- Test error decreases even after training error reached zero!

Resistance to Overfitting

- Often we observe AdaBoost behaving like this:

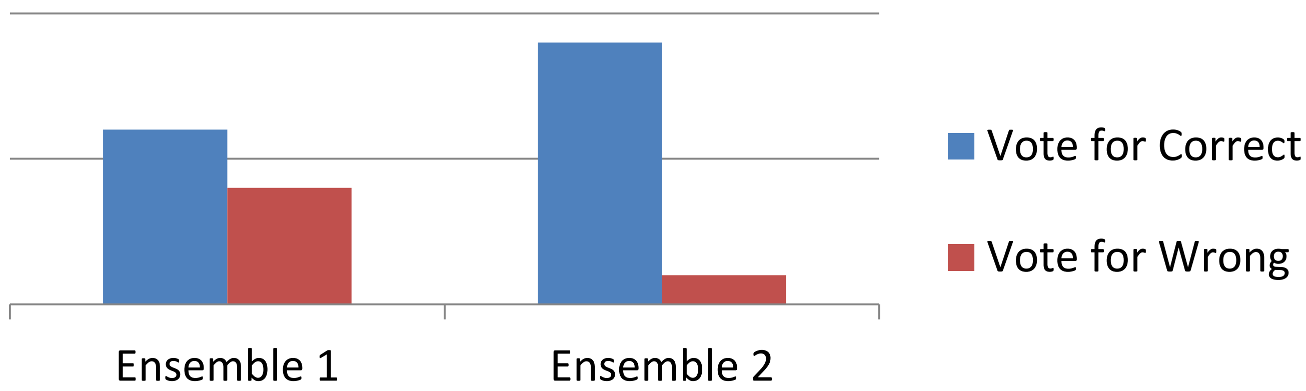


Notice something strange?

- Test error decreases even after training error reached zero!
- But… more weak learners… more complex hypothesis, how come we are not overfitting?

An Explanation: Margin Theory

- **Voting Margin = Fraction Voting Correctly – Fraction Voting Incorrectly**
- **Not SVM margin**, but related; **measures confidence** of the ensemble



- AdaBoost keeps increasing the margins even after it has managed to classify all training examples correctly (Schapire et al, 1998)

Interpretations of AdaBoost

- PAC learning (Schapire, 1990)
- Game Theory (Freund & Schapire, 1996)
- VC-Theory (Freund & Schapire, 1997)
- **Margin Theory** (Schapire et al, 1998)
- Information Theory (Kivinen & Warmuth, 1999)
- Optimizing Bregman Divergence (Collins, 2000)
- Minimization of an exponential loss (Friedman et al, 2000)
- Functional Gradient Descent (Mason et al, 2001)
- Probabilistic (Lebanon & Lafferty, 2001; Edakunni, Brown & Kovacs, 2011)
- Dynamical systems (Rudin et al, 2004)
- Implicit regularization via early stopping (Rosset et al, 2004; Zhao & Yu, 2007)

- And many more, mostly **complementary**; each explains some aspects

Strengths & Weaknesses

Strengths
Few parameters
Simple to implement
Implicit feature selection
Resistant to overfitting (when low noise)
Performs very well in practice

Strengths & Weaknesses

Strengths	Weaknesses
Few parameters	**Needs a termination condition**
Simple to implement	**Sensitive to noisy data & outliers Why?**
Implicit feature selection	**Must adjust for cost-sensitive or imbalanced class problems**
Resistant to overfitting (when low noise)	**Must adjust to handle multiclass tasks**
Performs very well in practice	

What we Learned

- Boosting: **Turning a weak learner into a strong one**
- AdaBoost **powerful** and **popular ensemble** method
- **Consistently ranks well** w.r.t. other learning algorithms
- **Each round focus on examples previously misclassified**
- **Different** than **Bagging**
- **Strengths**: simple, few parameters, implicit feature selection, resistant to overfitting (expl. by margin theory)
- **Weaknesses**: outliers/noise, termination, skew/cost-insensitive, must be modified for multiclass problems
- Many interpretations, a **fertile research area**

Thank You!

Questions?