A Theory of Multiclass Boosting

Indraneel Mukherjee*, R. E. Schapire
Princeton University

Wrigley Field prepared for college **football** game

Sports

Dublin warned over ECB **liquidity**

Business

Newest **senators** Coons and Manchin sworn in

Politics

Wrigley Field prepared for college **football** game

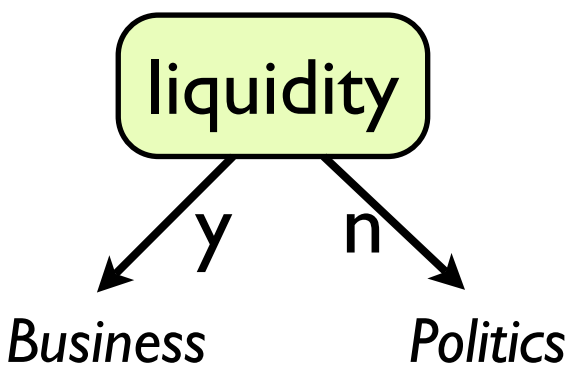
Sports

Dublin warned over ECB **liquidity**

Business

Newest **senators** Coons and Manchin sworn in

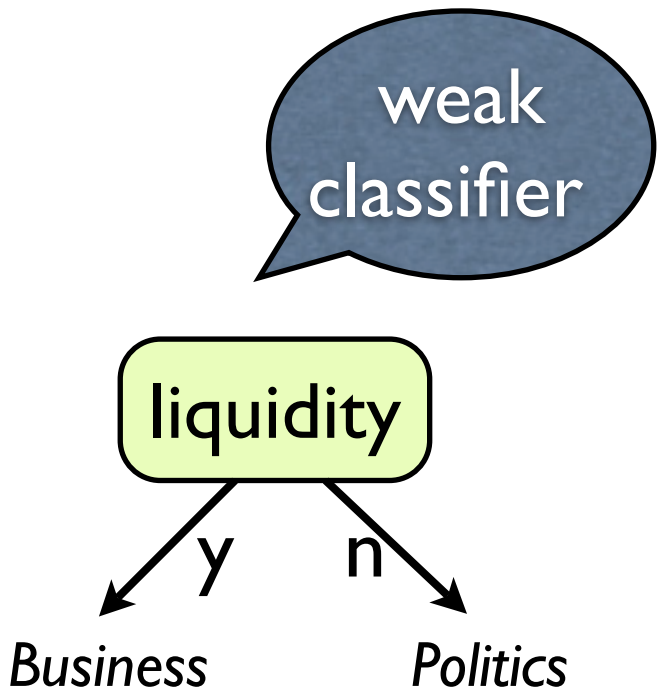
Politics



Wrigley Field prepared for college **football** game — *Sports*

Dublin warned over ECB **liquidity** — *Business*

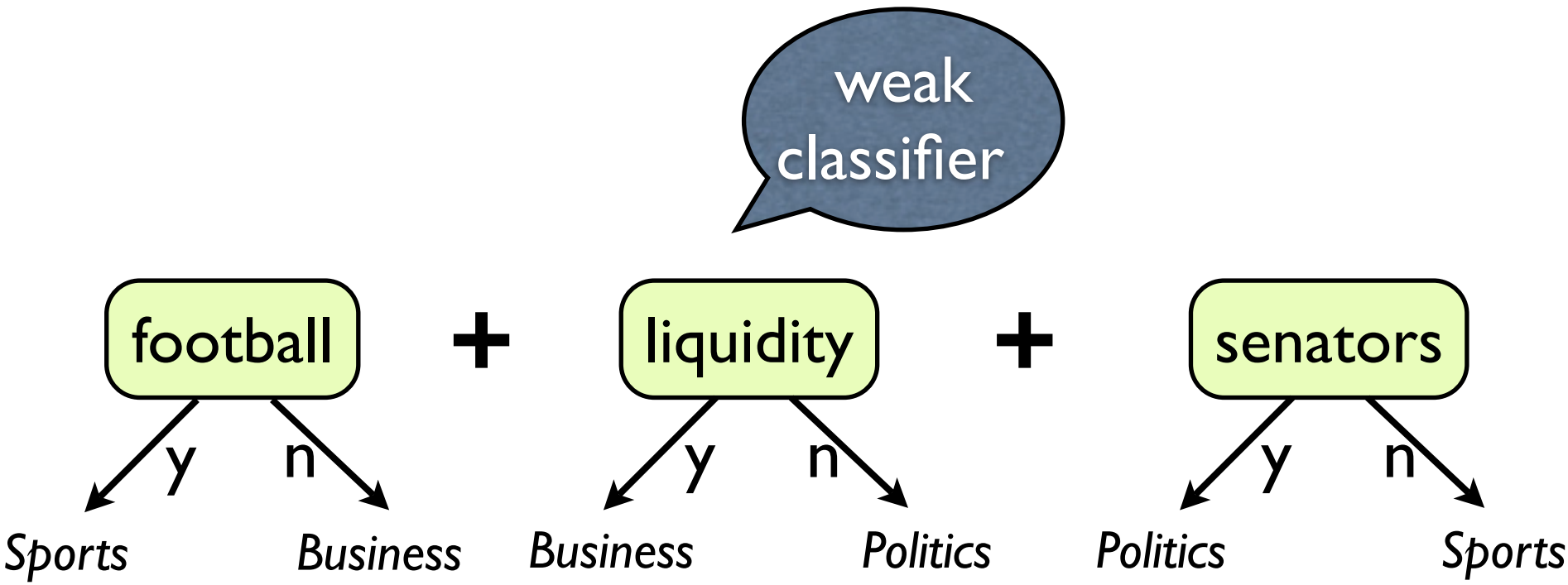
Newest **senators** Coons and Manchin sworn in — *Politics*



Wrigley Field prepared for college **football** game — *Sports*

Dublin warned over ECB **liquidity** — *Business*

Newest **senators** Coons and Manchin sworn in — *Politics*



Goals of Boosting

Goals of Boosting

- Boost *simplest* weak classifiers

Goals of Boosting

- Boost *simplest* weak classifiers
  - Use right *weak learning condition (WLC)*

Goals of Boosting

- Boost *simplest* weak classifiers
  - Use right *weak learning condition (WLC)*
- Important for generalization error:

Goals of Boosting

- Boost *simplest* weak classifiers
  - Use right *weak learning condition (WLC)*
- Important for generalization error:
  - Simple weak classifier may imply less overfitting

Goals of Boosting

- Boost *simplest* weak classifiers
  - Use right *weak learning condition (WLC)*
- Important for generalization error:
  - Simple weak classifier may imply less overfitting
  - Too simple could lead to underfitting

Goals of Boosting

- Boost *simplest* weak classifiers
  - Use right *weak learning condition (WLC)*
- Important for generalization error:
  - Simple weak classifier may imply less overfitting
  - Too simple could lead to underfitting
- Theory known for binary, not for multiclass

This Talk

This Talk

- Existing frameworks inadequate for multiclass

This Talk

- Existing frameworks inadequate for multiclass
  - Most resulting WLC's are *too weak* or *too strong*

This Talk

- Existing frameworks inadequate for multiclass
  - Most resulting WLC's are *too weak* or *too strong*
- Introduce *new framework for multiclass boosting*

This Talk

- Existing frameworks inadequate for multiclass
  - Most resulting WLC's are *too weak* or *too strong*
- Introduce *new framework for multiclass boosting*
  - Captures the *minimal WLC*

This Talk

- Existing frameworks inadequate for multiclass
  - Most resulting WLC's are *too weak* or *too strong*
- Introduce *new framework for multiclass boosting*
  - Captures the *minimal WLC*
- *Boosting algorithm using the minimal WLC*

This Talk

- Existing frameworks inadequate for multiclass
  - Most resulting WLC's are *too weak* or *too strong*
- Introduce *new framework for multiclass boosting*
  - Captures the *minimal WLC*
- *Boosting algorithm using the minimal WLC*
  - Provably drives down error efficiently

This Talk

- Existing frameworks inadequate for multiclass
  - Most resulting WLC's are *too weak* or *too strong*
- Introduce *new framework for multiclass boosting*
  - Captures the *minimal WLC*
- *Boosting algorithm using the minimal WLC*
  - Provably drives down error efficiently
  - Experiments to complement the theory

Binary boosting

Binary boosting

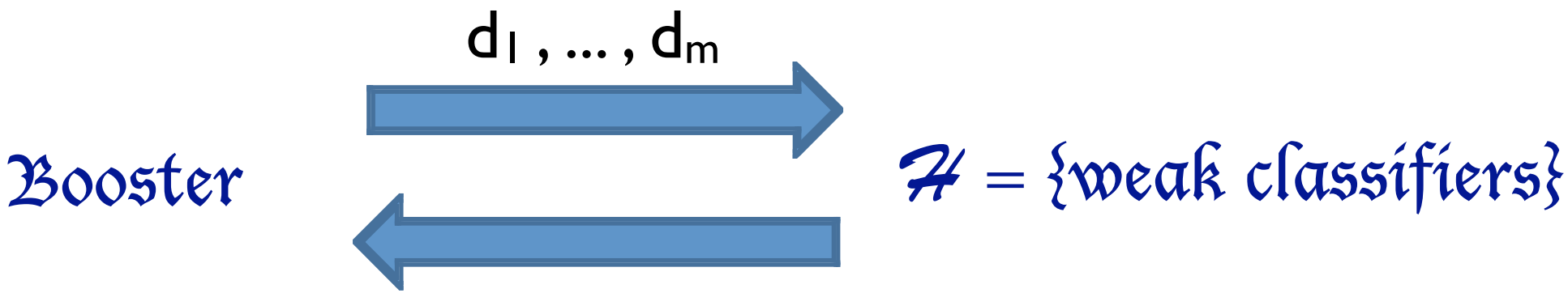
Input: (x_1, y_1) , ... , (x_m, y_m)

Booster

$\mathcal{H}$ = {weak classifiers}

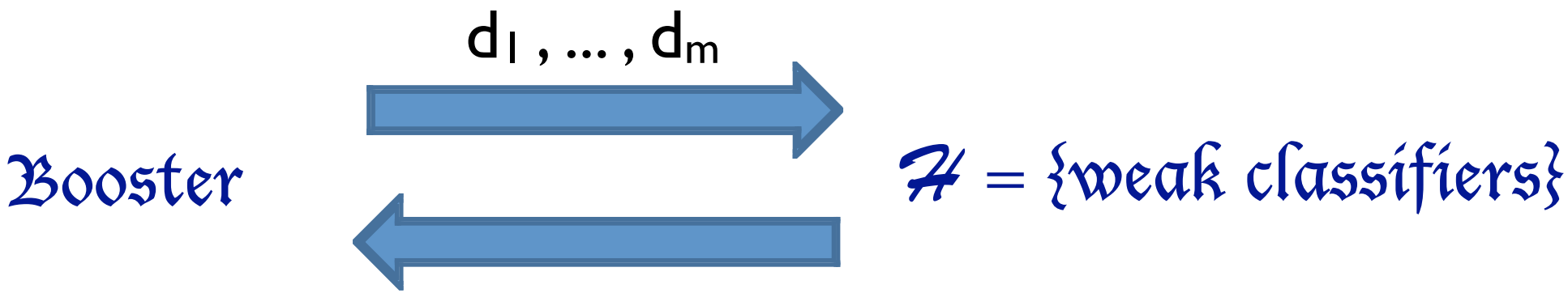
Binary boosting

Input: (x_1, y_1) , … , (x_m, y_m)



Binary boosting

Input: (x_1, y_1) , ... , (x_m, y_m)

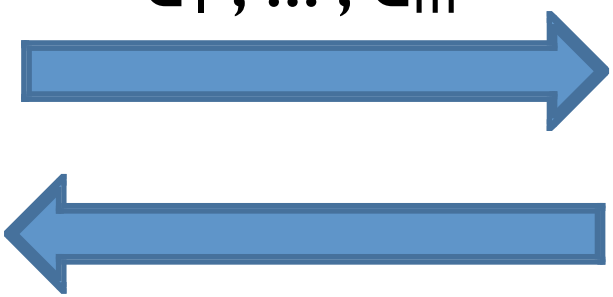


Condition : $\widehat{\text{err}}_{\mathbf{d}}(h) \leq \frac{1}{2} - \gamma$

Binary boosting

Input: (x_1,y_1) , … , (x_m, y_m)

d_1 , … , d_m



Booster

$\mathcal{H}$ = {weak classifiers}

$h \in \mathcal{H}$, h: {*Example*} ⇒ {*Label*}

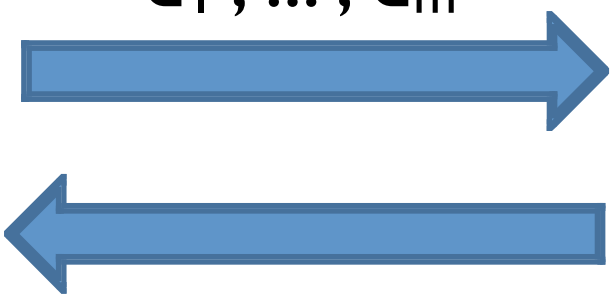
Condition : $\widehat{\text{err}}_d(h) \leq \frac{1}{2} - \gamma$

Binary WLC

Binary boosting

Input: (x_1, y_1) , ... , (x_m, y_m)

d_1 , ... , d_m



Booster

$\mathcal{H}$ = {weak classifiers}

$h \in \mathcal{H}$, h: {*Example*} ⇒ {*Label*}

Condition : $\widehat{\text{err}}_d(h) \leq \frac{1}{2} - \gamma$

Binary WLC

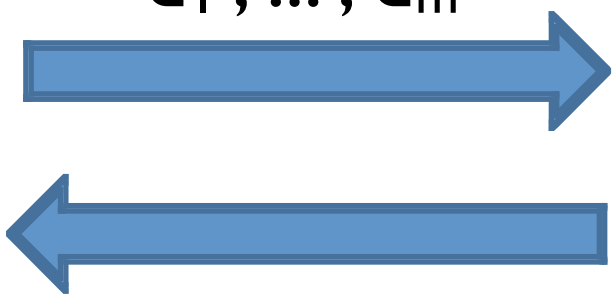
Final model: (weighted) majority$\{h_1, \ldots, h_T\}$

Binary boosting

More weight on misclassified examples

Input: (x_1, y_1) , ... , (x_m, y_m)

d_1 , ... , d_m



Booster

$\mathcal{H}$ = {weak classifiers}

$h \in \mathcal{H}$, h: {*Example*} ⇒ {*Label*}

Condition : $\widehat{\text{err}}_d(h) \leq \frac{1}{2} - \gamma$

Binary WLC

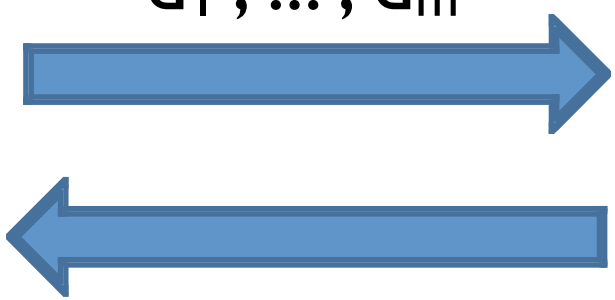
Final model: (weighted) majority$\{h_1, \ldots, h_T\}$

Binary boosting

More weight on misclassified examples

Input: (x_1, y_1) , ... , (x_m, y_m)

d_1 , ... , d_m



Booster

$\mathcal{H}$ = {weak classifiers}

$h \in \mathcal{H}$, h: {*Example*} ⇒ {*Label*}

Condition : $\widehat{\text{err}}_d(h) \leq \frac{1}{2} - \gamma$

Binary WLC

Final model: (weighted) majority$\{h_1, \ldots, h_T\}$

After T rounds, $\widehat{\text{err}}$ of maj$\{h_1, \ldots, h_T\} \leq \exp(-T\gamma^2/2)$

Binary WLC Ideal

Binary WLC Ideal

- *Required tasks easy*. Only better than random

Binary WLC Ideal

- *Required tasks easy*. Only better than random
- *Sufficient.* $\mathcal{H}$ satisfies binary WLC => $\mathcal{H}$ is *boostable*

Binary WLC Ideal

- *Required tasks easy*. Only better than random
- *Sufficient.* $\mathcal{H}$ satisfies binary WLC => $\mathcal{H}$ is *boostable*
  - *Boostable space*: contains perfect combination

Binary WLC Ideal

- *Required tasks easy*. Only better than random
- *Sufficient.* $\mathcal{H}$ satisfies binary WLC => $\mathcal{H}$ is *boostable*
  - *Boostable space*: contains perfect combination
- *Necessary*. Boostable space satisfies binary WLC

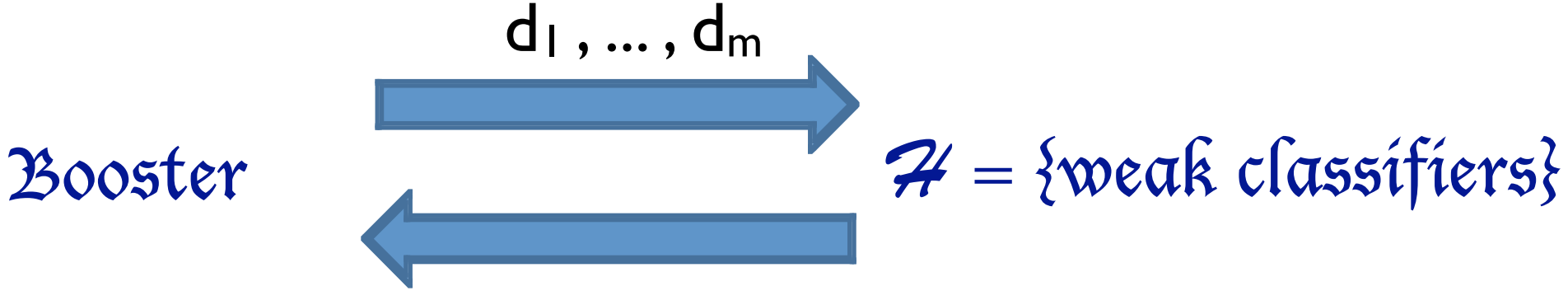
Binary WLC Ideal

- *Required tasks easy*. Only better than random
- *Sufficient.* $\mathcal{H}$ satisfies binary WLC => $\mathcal{H}$ is *boostable*
  - *Boostable space*: contains perfect combination
- *Necessary*. Boostable space satisfies binary WLC
- *Effective*. Allows efficient boosting algorithm

Extending to Multiclass

Extending to Multiclass

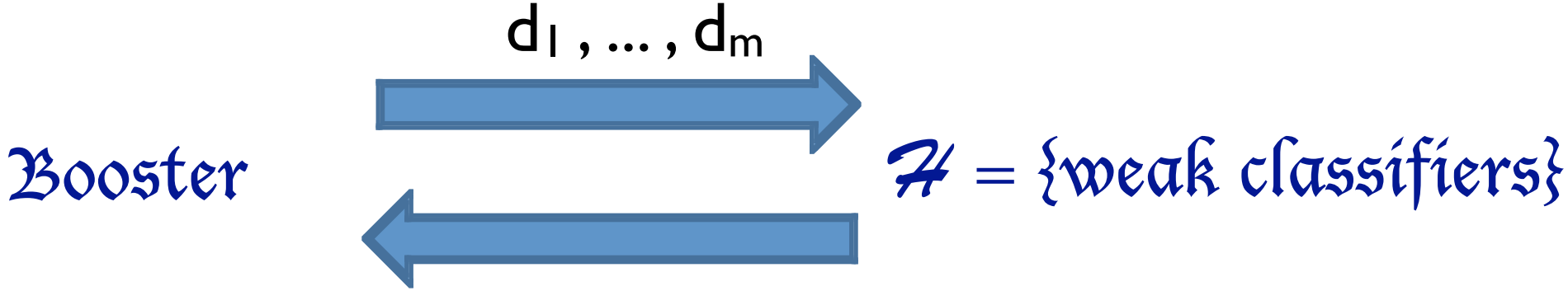
Input: (x_1, y_1) , … , (x_m, y_m)



$h \in \mathcal{H}$, h: {*Example*} ⇒ {*Multiclass Label*}

Extending to Multiclass

Input: (x_1, y_1) , ... , (x_m, y_m)



$h \in \mathcal{H}$, h: {*Example*} ⇒ {*Multiclass Label*}

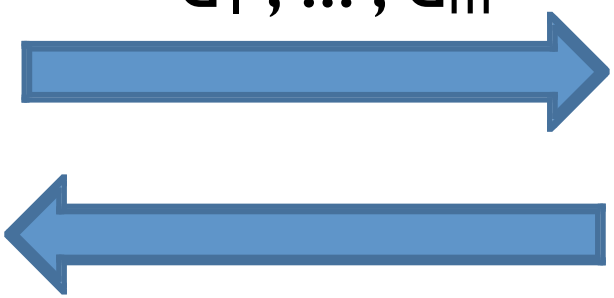
$$\widehat{\text{err}}_{\mathbf{d}}(h) \leq 1 - \frac{1}{k} - \gamma$$

SAMME [Zhu, Zou, Rosset, Hastie '09]

Extending to Multiclass

Input: (x_1, y_1) , … , (x_m, y_m)

d_1 , … , d_m



Booster

$\mathcal{H}$ = {weak classifiers}

$h \in \mathcal{H}$, h: {*Example*} ⇒ {*Multiclass Label*}

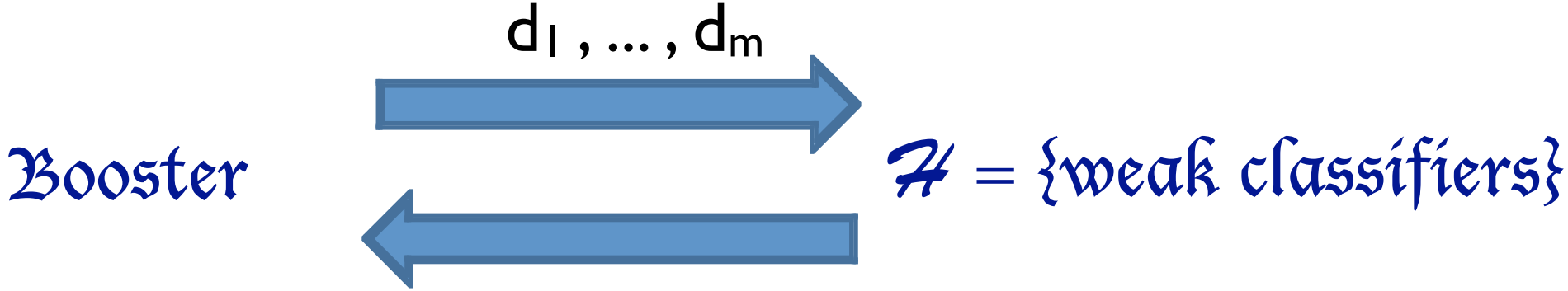
$$\widehat{\text{err}}_d(h) \leq 1 - \frac{1}{k} - \gamma$$

Too weak

SAMME [Zhu, Zou, Rosset, Hastie '09]

Extending to Multiclass

Input: (x_1, y_1) , … , (x_m, y_m)



$h \in \mathcal{H}$, h: {*Example*} ⇒ {*Multiclass Label*}

$$\widehat{\text{err}}_{\mathbf{d}}(h) \leq 1 - \frac{1}{k} - \gamma$$

Too weak

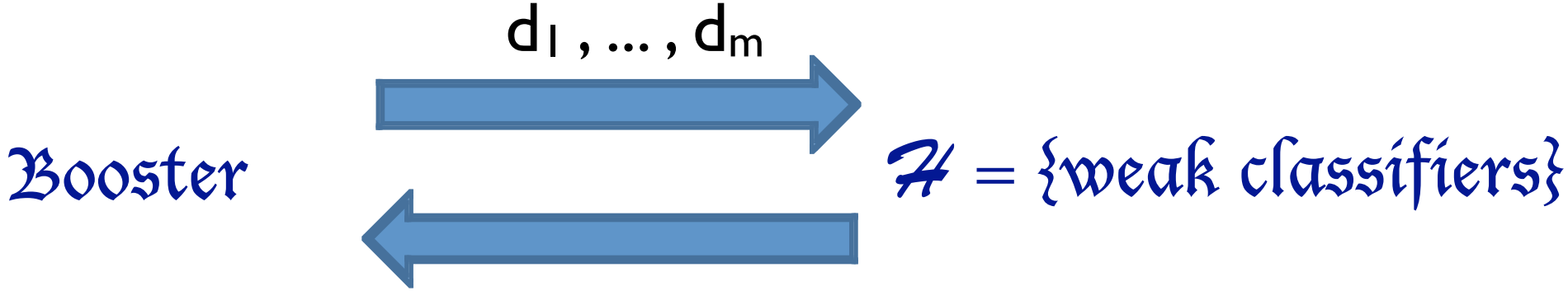
SAMME [Zhu, Zou, Rosset, Hastie '09]

$$\widehat{\text{err}}_{\mathbf{d}}(h) \leq \frac{1}{2} - \gamma$$

AdaBoost.M1 [Freund, Schapire '96]

Extending to Multiclass

Input: (x_1, y_1) , … , (x_m, y_m)



$h \in \mathcal{H}$, h: {*Example*} ⇒ {*Multiclass Label*}

$\widehat{\text{err}}_d(h) \leq 1 - \frac{1}{k} - \gamma$ — *Too weak*

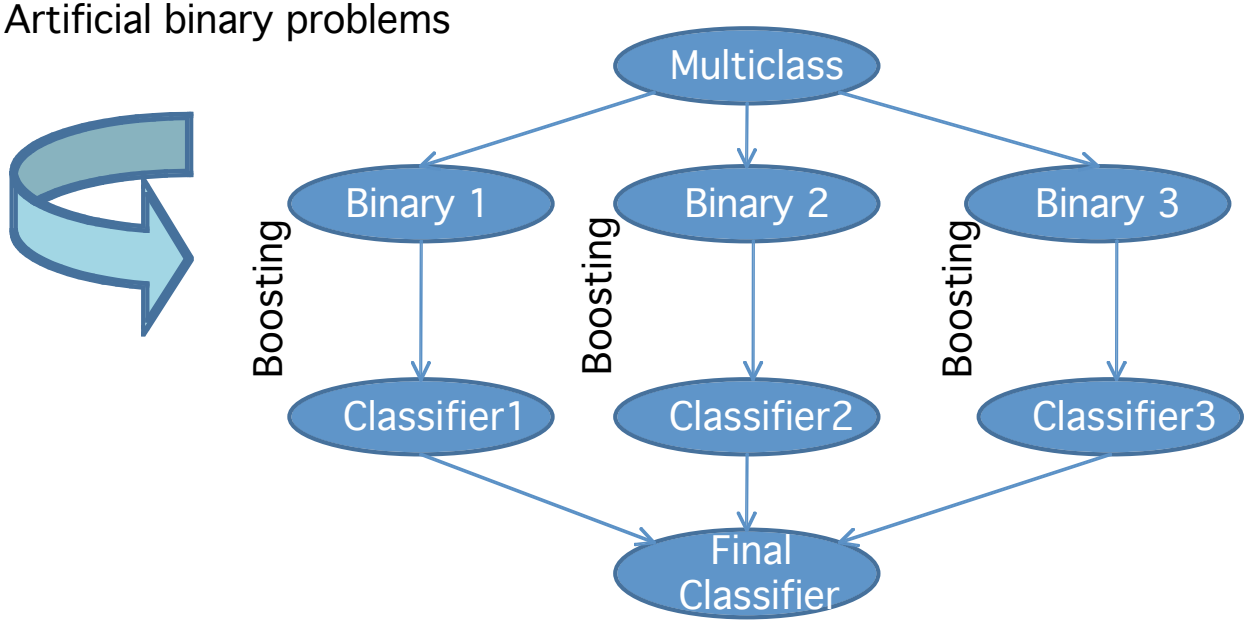
SAMME [Zhu, Zou, Rosset, Hastie '09]

$\widehat{\text{err}}_d(h) \leq \frac{1}{2} - \gamma$ — *Too strong*

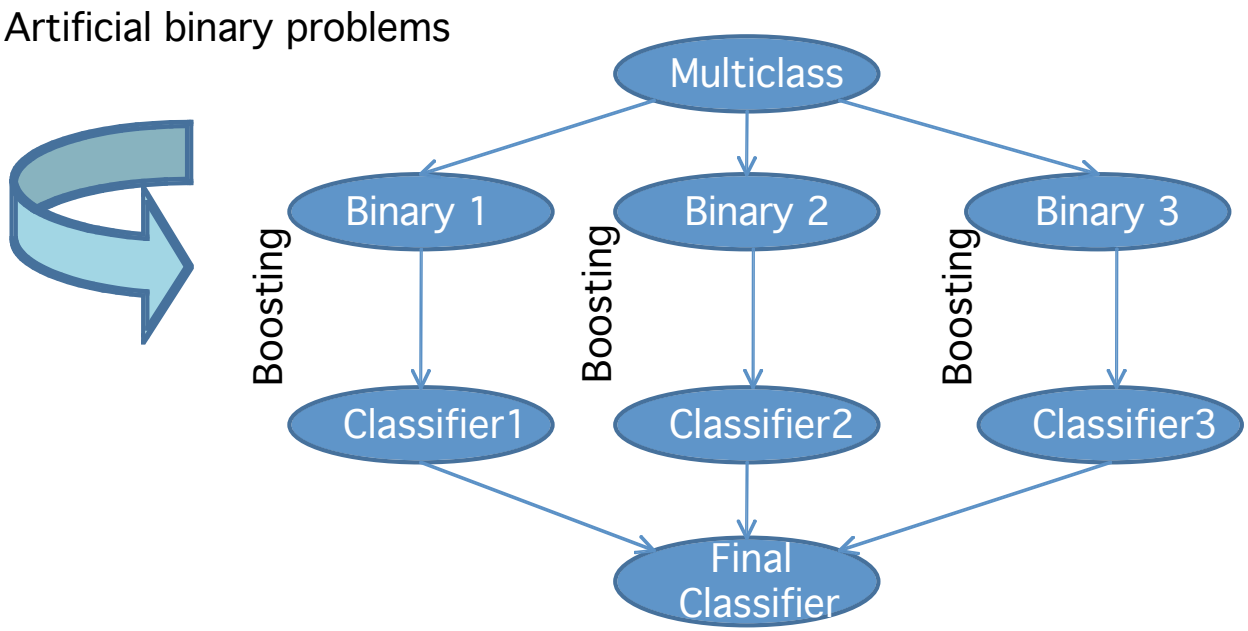
AdaBoost.M1 [Freund, Schapire '96]

Reduction to binary

Artificial binary problems

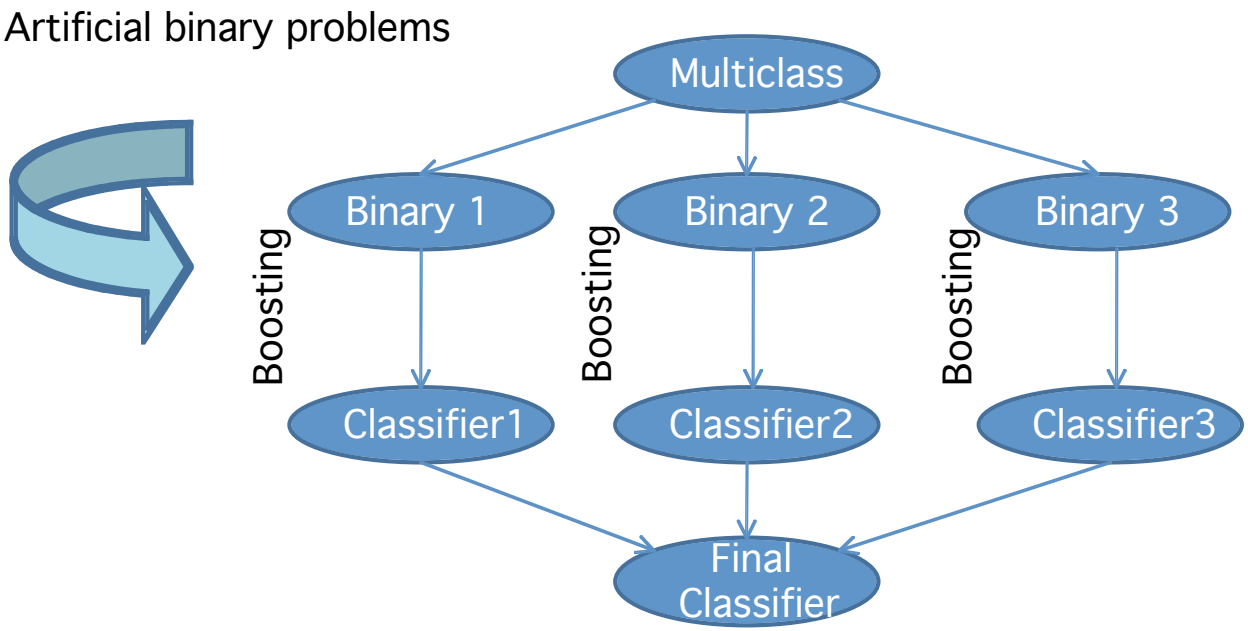


Reduction to binary



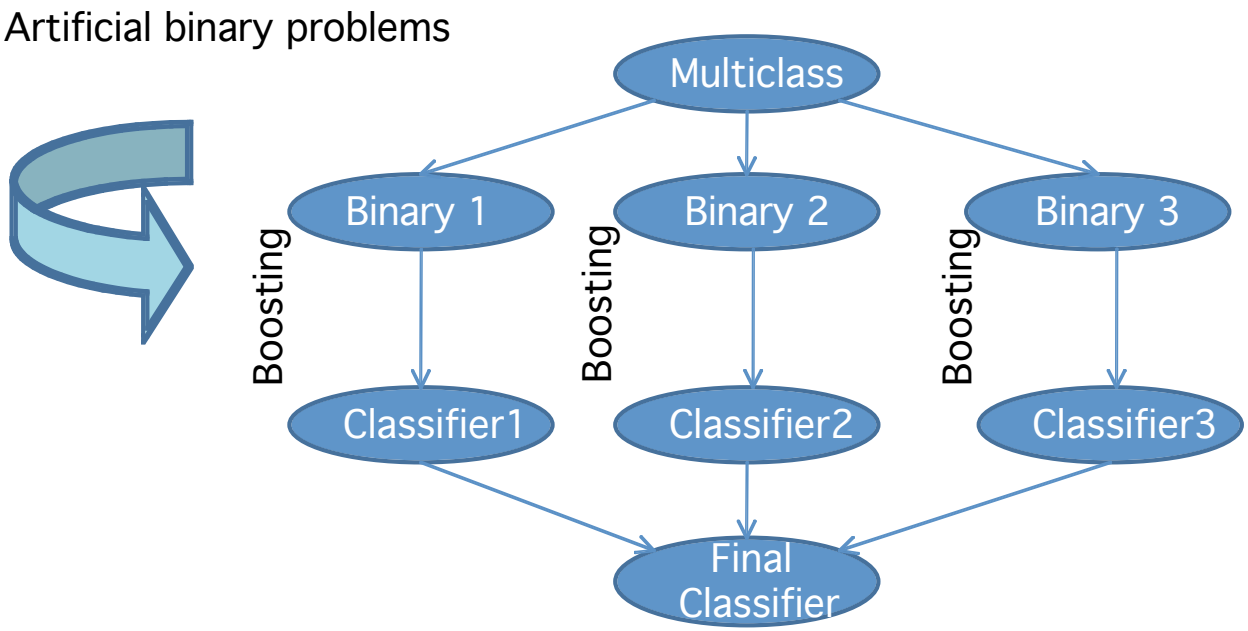
- One-against-all, all-pairs, ECOC. E.g.

Reduction to binary



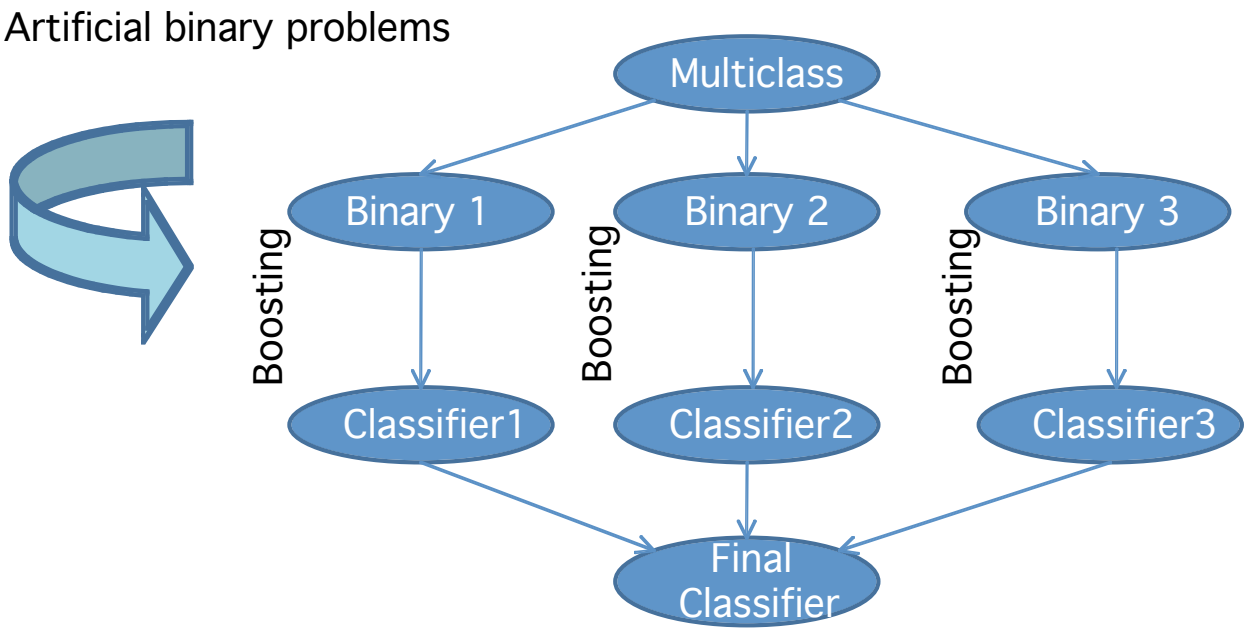
- One-against-all, all-pairs, ECOC. E.g.
  - One-against-all: (AdaBoost.MH)[Schapire & Singer '99]

Reduction to binary



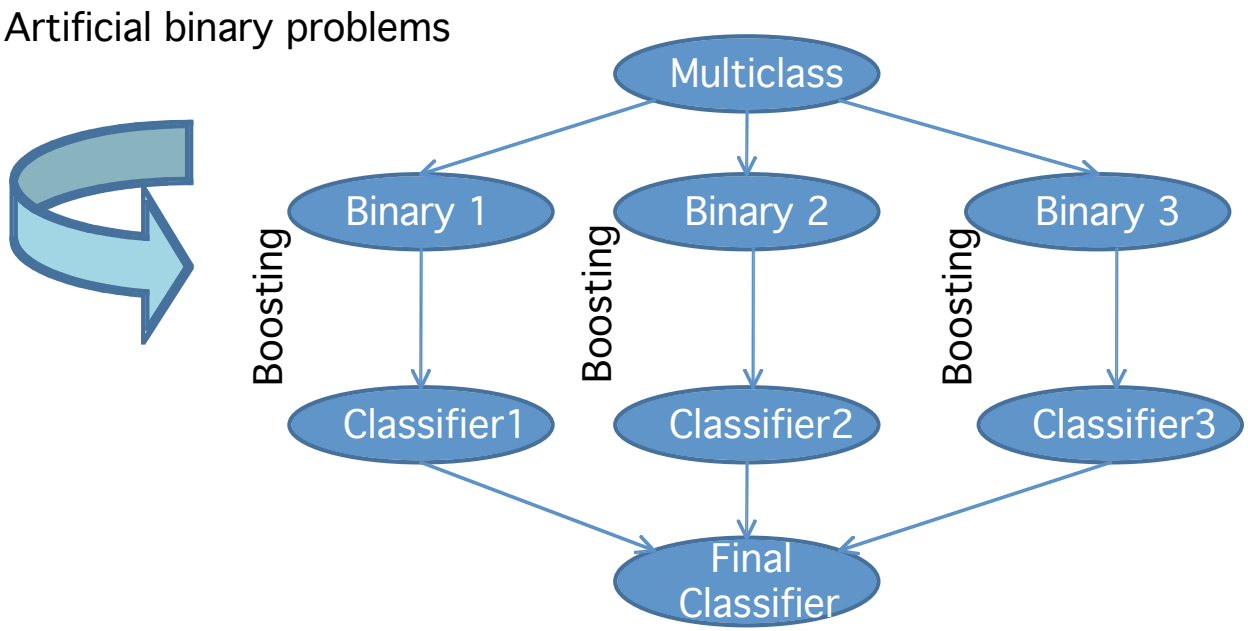
- One-against-all, all-pairs, ECOC. E.g.
  - One-against-all: (AdaBoost.MH)[Schapire & Singer '99]
  - All-pairs: (AdaBoost.MR)[Freund & Schapire '96, Schapire & Singer '99]

Reduction to binary



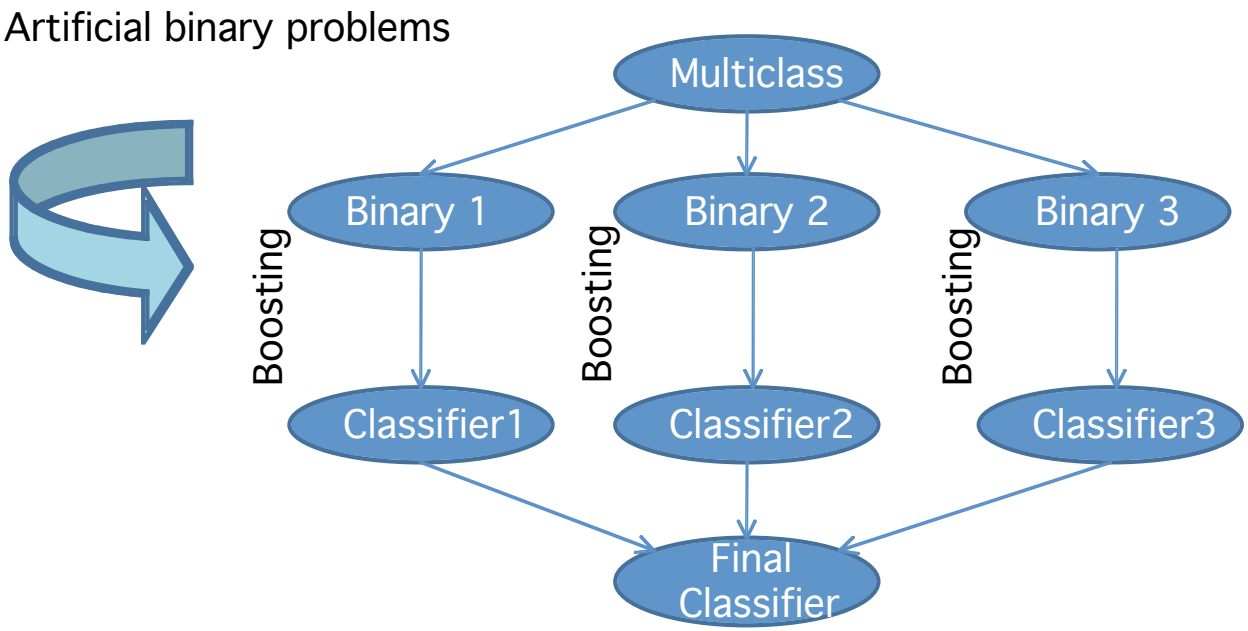
- One-against-all, all-pairs, ECOC. E.g.
  - One-against-all: (AdaBoost.MH)[Schapire & Singer '99]
  - All-pairs: (AdaBoost.MR)[Freund & Schapire '96, Schapire & Singer '99]
- Practical, but poorly understood

Reduction to binary



- One-against-all, all-pairs, ECOC. E.g.
  - One-against-all: (AdaBoost.MH)[Schapire & Singer '99]
  - All-pairs: (AdaBoost.MR)[Freund & Schapire '96, Schapire & Singer '99]
- Practical, but poorly understood
- Sometimes *too strong*

Reduction to binary



- One-against-all, all-pairs, ECOC. E.g.
  - One-against-all: (AdaBoost.MH)[Schapire & Singer '99]
  - All-pairs: (AdaBoost.MR)[Freund & Schapire '96, Schapire & Singer '99]
- Practical, but poorly understood
- Sometimes *too strong*
  - e.g. One-against-all (AdaBoost.MH)

New Framework *(simplified)*

New Framework *(simplified)*

- Booster sends *cost matrix C*, not distribution

New Framework *(simplified)*

- Booster sends *cost matrix C*, not distribution
  - C(i, ℓ): cost of predicting label ℓ on example i
  - Cost(C, h) = $\sum_i$ C(i, h(x_i))

New Framework *(simplified)*

- Booster sends *cost matrix C*, not distribution
  - C(i, ℓ): cost of predicting label ℓ on example i
  - Cost(C, h) $= \sum_i$ C(i, h(x_i))
- Perform as well as fixed *baseline* predictor *B*

New Framework *(simplified)*

- Booster sends *cost matrix C*, not distribution
  - C(i, ℓ): cost of predicting label ℓ on example i
  - Cost(C, h) $= \sum_i$ C(i, h(x_i))
- Perform as well as fixed *baseline* predictor *B*
  - B(i, ℓ): prob. with which B predicts ℓ on i
  - Cost(C, B) $= \sum_i \mathbb{E}$[C(i, B(x_i))] $= \sum_i \sum_\ell$ C(i, l) B(i, ℓ)

New Framework *(simplified)*

- Booster sends *cost matrix C*, not distribution
  - C(i, ℓ): cost of predicting label ℓ on example i
  - Cost(C, h) $= \sum_i$ C(i, h(x_i))
- Perform as well as fixed *baseline* predictor *B*
  - B(i, ℓ): prob. with which B predicts ℓ on i
  - Cost(C, B) $= \sum_i \mathbb{E}$[C(i, B(x_i))] $= \sum_i \sum_\ell$ C(i, l) B(i, ℓ)
- *Restriction*: Cost (C, h) ≤ Cost (C, B)

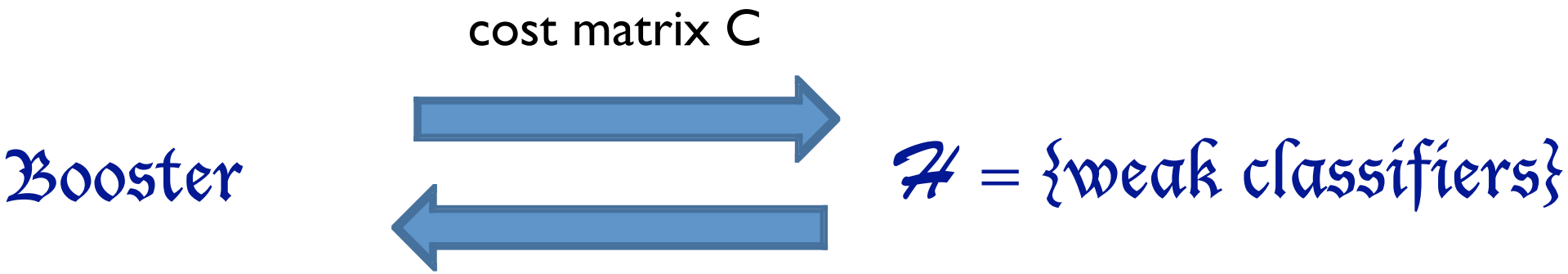
New Framework *(simplified)*

New Framework *(simplified)*

Parameter: Fixed baseline B

New Framework *(simplified)*

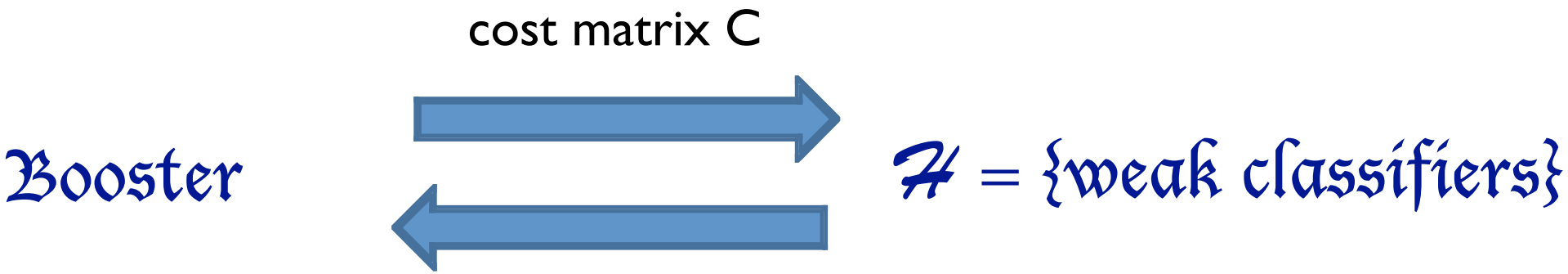
Parameter: Fixed baseline B



h ∈ $\mathcal{H}$, *h*: {*Example*} ⇒ {*Label*}

New Framework *(simplified)*

Parameter: Fixed baseline B

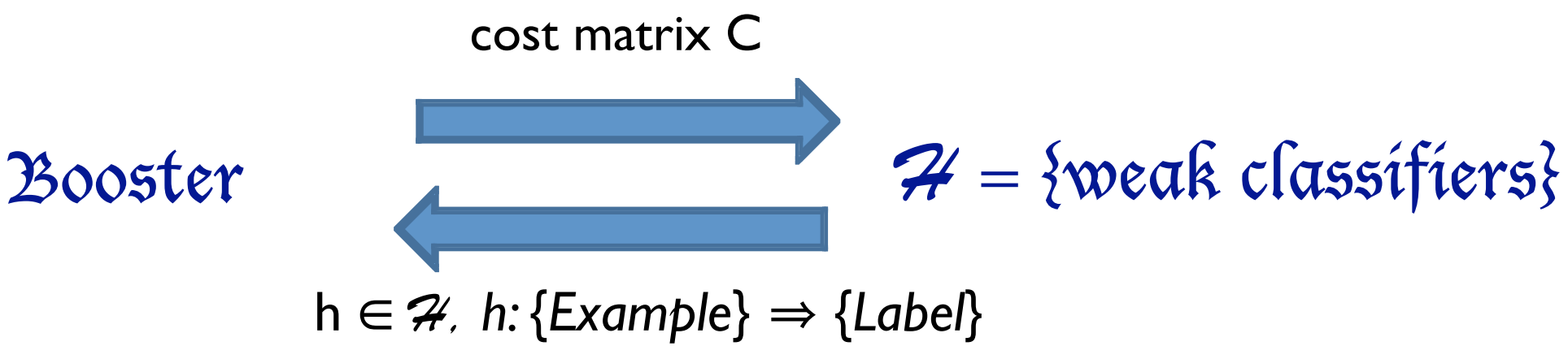


h ∈ $\mathcal{H}$, *h*: {*Example*} ⇒ {*Label*}

Cost(C, h) ≤ Cost(C, B)

Binary Boosting

$$B(i, \ell) = \begin{cases} \frac{1}{2} + \gamma & \text{if } \ell \text{ correct} \\ \frac{1}{2} - \gamma & \text{if } \ell \text{ wrong} \end{cases}$$



$$\text{Cost}(C, h) \leq \text{Cost}(C, B)$$

Edge-over-random WLC

Edge-over-random WLC

- Edge-over-random baseline Q

Edge-over-random WLC

- Edge-over-random baseline Q
  - B(i, correct) $\geq$ B(i, wrong) + 2γ
  - B(i, $\cdot$) is a distribution

Edge-over-random WLC

- Edge-over-random baseline Q
  - B(i, correct) $\geq$ B(i, wrong) + 2γ
  - B(i, $\cdot$) is a distribution
- Many choices for B (only one for binary)

Edge-over-random WLC

- Edge-over-random baseline Q
  - B(i, correct) $\geq$ B(i, wrong) + 2γ
  - B(i, $\cdot$) is a distribution
- Many choices for B (only one for binary)
- *Condition with such baseline:*

Edge-over-random WLC

- Edge-over-random baseline Q
  - B(i, correct) $\geq$ B(i, wrong) + 2γ
  - B(i, $\cdot$) is a distribution
- Many choices for B (only one for binary)
- *Condition with such baseline:*

  Edge-over-random WLC

EOR nearly Ideal

EOR nearly Ideal

- *Required tasks easy.* Only beat random

EOR nearly Ideal

- *Required tasks easy.* Only beat random
- *Sufficient.* Satisfying EOR implies boostability

EOR nearly Ideal

- *Required tasks easy.* Only beat random
- *Sufficient.* Satisfying EOR implies boostability
- *Effective*. Allows efficient boosting

EOR nearly Ideal

- *Required tasks easy.* Only beat random
- *Sufficient.* Satisfying EOR implies boostability
- *Effective.* Allows efficient boosting
- *Not Necessary.* For any EOR *(B)*, there is some boostable space $\mathcal{H}$ that does not satisfy it.

EOR nearly Ideal

- *Required tasks easy.* Only beat random
- *Sufficient.* Satisfying EOR implies boostability
- *Effective.* Allows efficient boosting
- *Not Necessary.* For any EOR *(B)*, there is some boostable space $\mathcal{H}$ that does not satisfy it.
- *Relaxed necessity. For any boostable space $\mathcal{H}$, there is some EOR (B) that $\mathcal{H}$ satisfies*

EOR nearly Ideal

- *Required tasks easy.* Only beat random
- *Sufficient.* Satisfying EOR implies boostability
- *Effective.* Allows efficient boosting
- *Not Necessary.* For any EOR *(B)*, there is some boostable space $\mathcal{H}$ that does not satisfy it.
- *Relaxed necessity. For any boostable space $\mathcal{H}$, there is some EOR (B) that $\mathcal{H}$ satisfies*
- Combine to form *single minimal WLC*

EOR nearly Ideal

- *Required tasks easy.* Only beat random
- *Sufficient.* Satisfying EOR implies boostability
- *Effective.* Allows efficient boosting
- *Not Necessary.* For any EOR *(B)*, there is some boostable space $\mathcal{H}$ that does not satisfy it.
- *Relaxed necessity. For any boostable space $\mathcal{H}$, there is some EOR (B) that $\mathcal{H}$ satisfies*
- Combine to form *single minimal WLC*
  - *Necessary and sufficient for boostability*

Boosting Algorithms

Boosting Algorithms

- *Optimally efficient* algorithm for any fixed EOR

Boosting Algorithms

- *Optimally efficient* algorithm for any fixed EOR
  - Like Boost-by-majority [Freund '95]

Boosting Algorithms

- *Optimally efficient* algorithm for any fixed EOR
  - Like Boost-by-majority [Freund '95]
  - Non-adaptive. Requires knowledge of γ

Boosting Algorithms

- *Optimally efficient* algorithm for any fixed EOR
  - Like Boost-by-majority [Freund '95]
  - Non-adaptive. Requires knowledge of γ
- *Adaptive* algorithm *assuming the minimal WLC*

Boosting Algorithms

- *Optimally efficient* algorithm for any fixed EOR
  - Like Boost-by-majority [Freund '95]
  - Non-adaptive. Requires knowledge of γ
- *Adaptive* algorithm *assuming the minimal WLC*
  - Based on multiplicative updates, like AdaBoost

Boosting Algorithms

- *Optimally efficient* algorithm for any fixed EOR
  - Like Boost-by-majority [Freund '95]
  - Non-adaptive. Requires knowledge of γ
- *Adaptive* algorithm *assuming the minimal WLC*
  - Based on multiplicative updates, like AdaBoost
  - Not optimal, but still provably very efficient

Adaptive Algorithm

Adaptive Algorithm

- In each round t:

Adaptive Algorithm

- In each round t:
  - Create cost matrix C_t

Adaptive Algorithm

- In each round t:
  - Create cost matrix C_t
  - Receive weak classifier h_t with edge δ_t

Adaptive Algorithm

- In each round t:
  - Create cost matrix C_t
  - Receive weak classifier h_t with edge δ_t
  - Compute weight α_t and update $f_t = f_{t-1} + \alpha_t h_t$

Adaptive Algorithm

- In each round t:
  - Create cost matrix C_t
  - Receive weak classifier h_t with edge δ_t
  - Compute weight α_t and update $f_t = f_{t-1} + \alpha_t h_t$

Weight

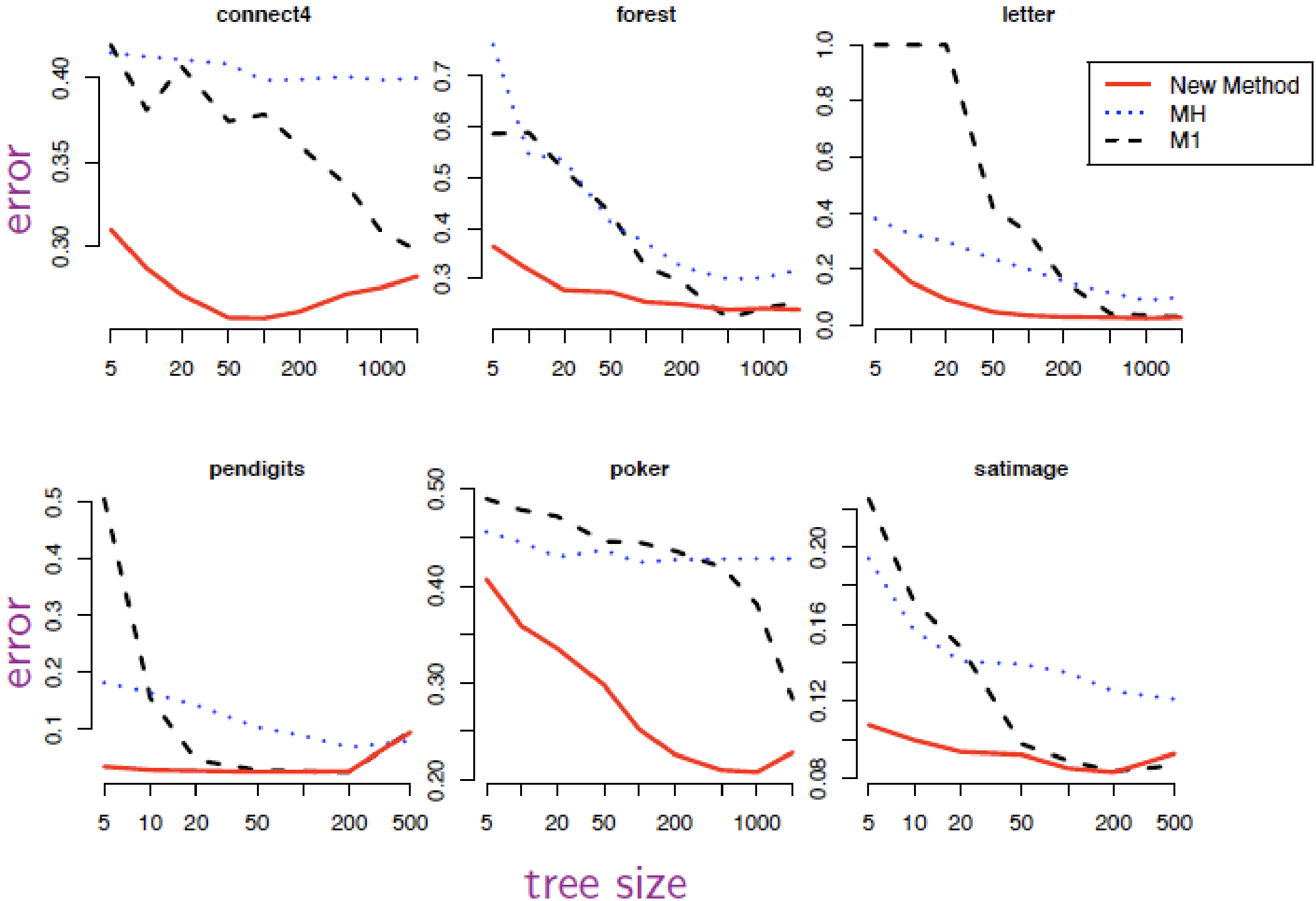
$$\alpha_t = \ln\left\{\frac{1+\delta_t}{1-\delta_t}\right\}$$

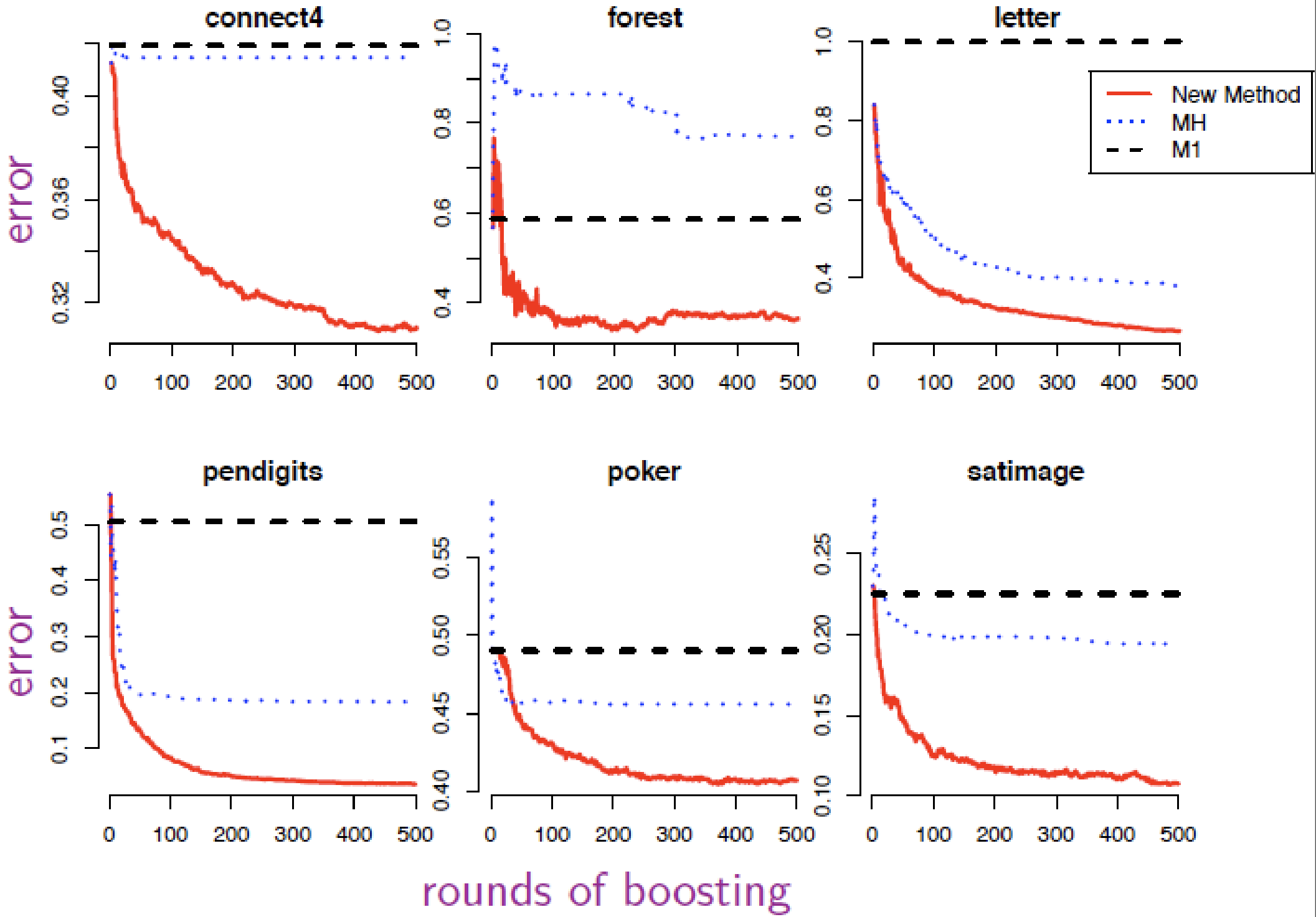
Cost Matrix

$$C_{t+1}(i,l) = \begin{cases} e^{f_t(i,l)-f_t(i,y_i)} & \text{if } l \neq y_i \\ -\sum_{l' \neq y_i} e^{f_t(i,l')-f_t(i,y_i)} & \text{if } l = y_i \end{cases}$$

Experiments

- Ran adaptive algorithm using minimal WLC
- Compared with AdaBoost.M1, AdaBoost.MH
- Tested on benchmark datasets
- Weak classifiers: bounded size decision trees


connect4
forest
letter
pendigits
poker
satimage
error
error
tree size
New Method
MH
M1
0.30
0.35
0.40
0.3
0.4
0.5
0.6
0.7
0.0
0.2
0.4
0.6
0.8
1.0
0.1
0.2
0.3
0.4
0.5
0.20
0.30
0.40
0.50
0.08
0.12
0.16
0.20
5
10
20
50
200
500
1000


connect4
forest
letter
pendigits
poker
satimage
error
error
rounds of boosting
New Method
MH
M1
0.32
0.36
0.40
0.4
0.6
0.8
1.0
0.4
0.6
0.8
1.0
0.1
0.2
0.3
0.4
0.5
0.40
0.45
0.50
0.55
0.10
0.15
0.20
0.25
0
100
200
300
400
500

Future Work

Future Work

- What happens with multi-label / confidence rated weak classifiers?

Future Work

- What happens with multi-label / confidence rated weak classifiers?
- Consistency of the algorithms.

Future Work

- What happens with multi-label / confidence rated weak classifiers?
- Consistency of the algorithms.
- Extensions to ranking.

Thank you