AdaCost: Misclassification Cost-Sensitive Boosting

Fan, Stolfo, Zhang, and Chan

Proceedings of **ICML** 1999

Presented by:
Michael Green
Department of Computer Science
University of California, San Diego
mgreen@cs.ucsd.edu

Today's Focus

• Motivation for and description of AdaCost algorithm

• Derivation of upper bound on misclassification cost

• Discussion of the cost function (β) and the hypothesis weight (α) to reduce the upper bound

• Evaluation of AdaCost against AdaBoost on real-world and publicly available data sets

Overview

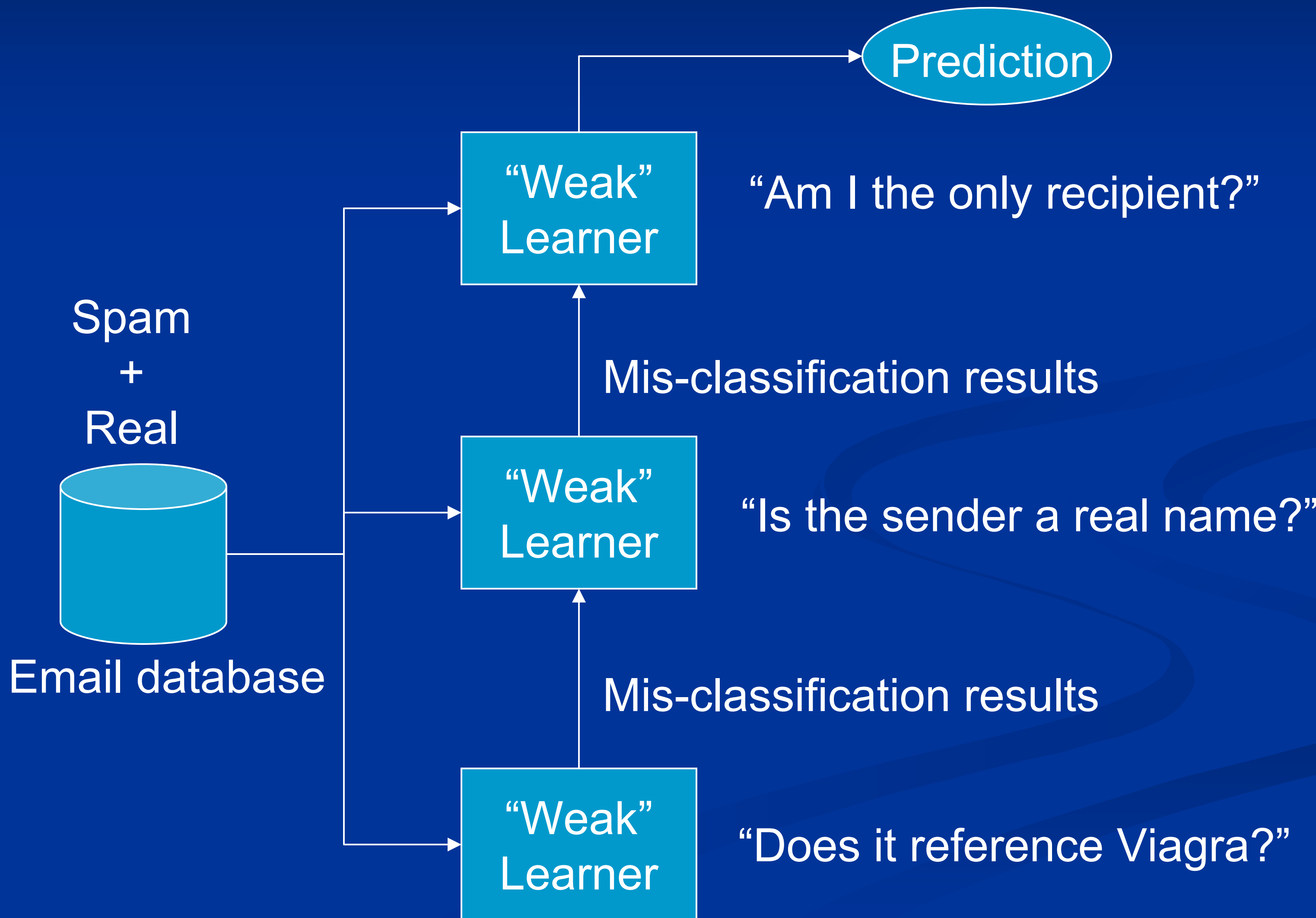
- Boosting methods provide a score, but assign equal weights to all classification errors.

- Misclassification of examples can have different costs (e.g., credit card fraud detection, cancer screening, etc.).

- AdaCost is a cost-sensitive boosting method intended to reduce the cumulative cost of misclassification.

- Experiments show potential for significant reduction in misclassification cost.

“weak” classifiers are “boosted”



Boosting Algorithm: AdaBoost

- Developed by Freund and Schapire in 1995

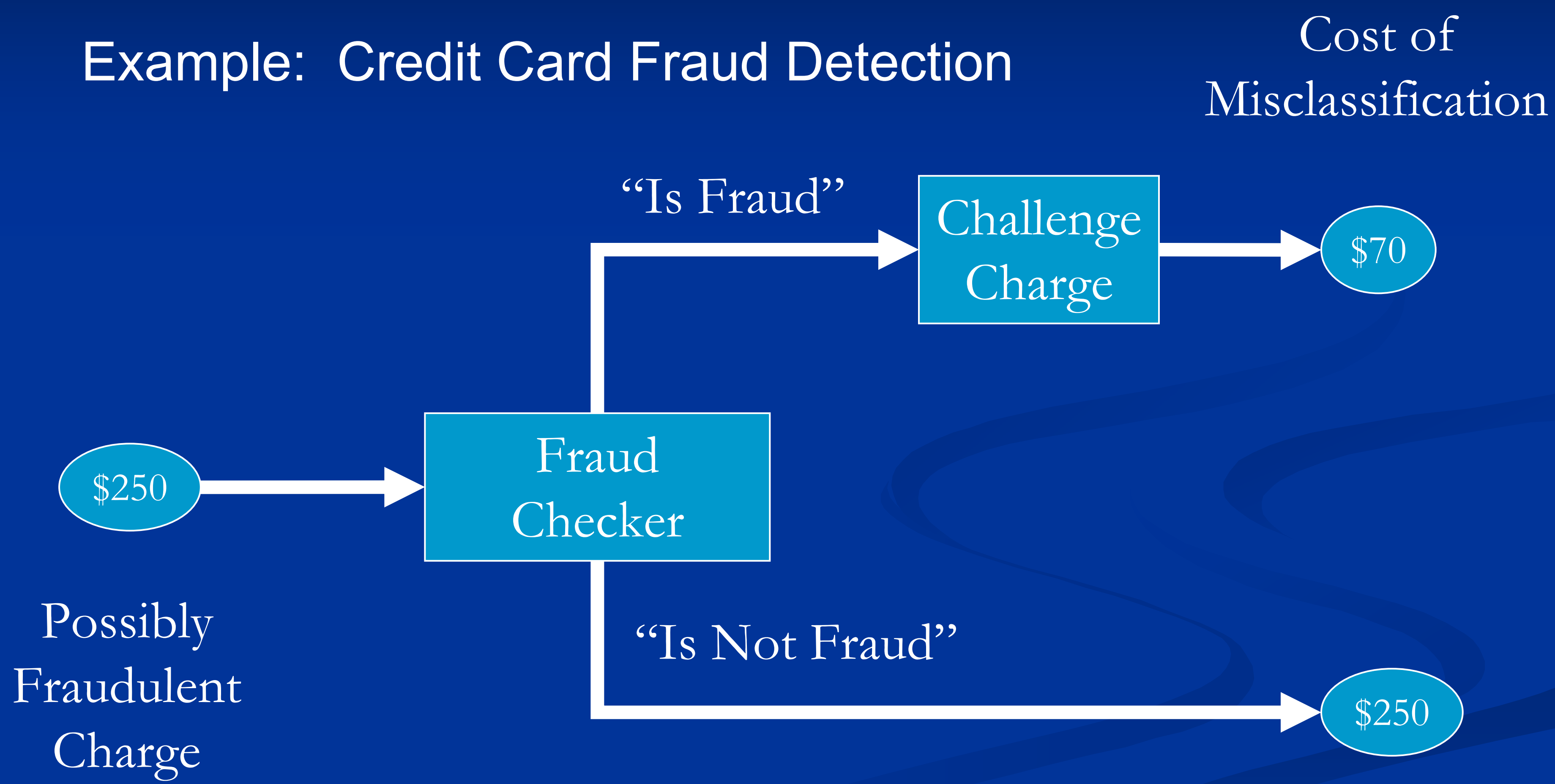
- Calls a given weak algorithm repeatedly in a series of rounds $t=1,\ldots,T$

- Weights of misclassified examples are increased in round $t+1$

- Issue: What if the cost of misclassifying an example differs depending on the type of misclassification?

Misclassification Costs

Example: Credit Card Fraud Detection



Cost-Sensitive Boosting

- Incorporates the cost of misclassified examples into the new weights for subsequent rounds

- Each round of the “weak” hypothesis targets the most expensive misclassifications of the previous round

- Assumes the user knows the costs of misclassification for each example

AdaCost in Particular

- Extends AdaBoost by incorporating a misclassification function, β, into the weight redistribution formula
- Allows for each example in the training set to have a different cost
- Each hypothesis outputs a prediction (a discrete label) and a confidence (a score)

AdaCost Algorithm Variables

- $S = \{(x_1,c_1,y_1),\ldots(x_m,c_m,y_m)\}$: a set of training examples
  - x_i in S belongs to a domain *X*
  - c_i cost factor belongs to non-negative reals
  - y_i in S belongs to finite label space *Y*
- t: index of the round of boosting
- h_t: weak hypothesis of form h: X->R (R is reals)
- $D_t(i)$: weight given to (x_i,c_i,y_i) in t-th round
- α_t: weight given to weak hypothesis at the t-th
- β(): cost adjustment function

AdaCost Algorithm

- Given: $S = \{(x_1, c_1, y_1), \ldots, (x_m, c_m, y_m)\}$; $x_i \in \mathcal{X}, c_i \in \mathbb{R}^+, y_i \in \{-1, +1\}$.
- Initialize $D_1(i)$ (such as $D_1(i) = c_i / \sum_j^m c_j$).
- For $t = 1, \ldots, T$:
  1. Train weak learner using distribution D_t.
  2. Compute weak hypothesis $h_t : \mathcal{X} \to \mathbb{R}$.
  3. Choose $\alpha_t \in \mathbb{R}$ and $\beta(i) \in \mathbb{R}^+$.
  4. Update

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

  where $\beta(i) = \beta\left(\operatorname{sign}(y_i h_t(x_i)), c_i\right)$ is a cost-adjustment function. Z_t is a normalization factor chosen so that D_{t+1} will be a distribution.

- Output the final hypothesis:

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

The Difference between AdaCost and AdaBoost

β increases weights more for each costly misclassified example, but <u>decreases</u> weight less otherwise

- $\beta(neg) \geq 0$, $\beta(pos) \geq 0$, $\alpha_t \geq 0$
- $\beta(neg)$ is non-decreasing w.r.t. c_i
- $\beta(pos)$ is non-increasing w.r.t. c_i

Misclassification Cost Upper Bound

Lemma 1

Let $f'(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \beta\Big(sign(yh_t(x)), c\Big)$ *and* $H'(x) = sign(f'(x))$. *If* $\forall c, \beta_-(c) \geq \beta_+(c)$, *the following is true:*

$$\forall x \in \mathcal{S} \; \Big(H'(x) = y \implies H(x) = y\Big)$$

Lemma Proof

Proof: By definition of H'(x) and f'(x):

$$H'(x) = y \Leftrightarrow yf'(x) = y \sum \alpha_t h_t(x) \beta(sign(yh_t(x)), c) > 0 \qquad (1)$$

…rewriting the right hand side as sum of correct and incorrect portions of predictions by the weak hypothesis

$$y \sum_{t_+} \alpha_{t_+} h_{t_+}(x) \beta_+ + y \sum_{t_-} \alpha_{t_-} h_{t_-}(x) \beta_- > 0 \qquad (2)$$

Lemma Proof

Since the lemma requires β(neg)≥ β(pos)≥0

$$\begin{aligned} y\sum_{t_+}\alpha_{t_+}h_{t_+}(x)\beta_+ \quad &+ \quad y\sum_{t_-}\alpha_{t_-}h_{t_-}(x)\beta_+ \\ &\geq \\ y\sum_{t_+}\alpha_{t_+}h_{t_+}(x)\beta_+ \quad &+ \quad y\sum_{t_-}\alpha_{t_-}h_{t_-}(x)\beta_- \end{aligned} \qquad (3)$$

Lemma Proof

Combining (2), (3), and the definition of f(x)

$$\beta_+(yf(x)) = y\sum_{t_+}\alpha_{t_+}h_{t_+}(x)\beta_+ + y\sum_{t_-}\alpha_{t_-}h_{t_-}(x)\beta_+ > 0 \qquad (4)$$

B(pos)>0 implies

$$yf(x) > 0 \qquad (5)$$

By definition of H(x) and f(x), (5) implies that H(x)=y

Theorem

The following holds for the upper bound of the training cumulative misclassification cost.

$$\sum c_i [\![H(x_i) \neq y_i]\!] \leq d \prod_{t=1}^{T} Z_t, \; d = \sum c_j$$

Proof

From the Lemma we know that

$$\sum c_i [\![H(x_i) \neq y_i]\!] \leq \sum c_i [\![H'(x_i) \neq y_i]\!] \qquad (6)$$

By unraveling the update rule [Schapire and Singer] we have

$$\begin{aligned} D_{T+1}(i) &= \frac{D_1(i)\exp(-\sum_t \alpha_t y_i h_t(x_i) \beta(i))}{\prod_t Z_t} \\ &= \frac{D_1(i)\exp(-y_i f'(x_i))}{\prod_t Z_t} \end{aligned} \qquad (7)$$

Proof

If $H'(x_i) \neq y_i$, then $y_i f'(x_i) \leq 0$ implying that $\exp(-y_i f'(x_i)) \geq 0$ and

$$[\![H'(x_i) \neq y_i]\!] \leq \exp(-y_i f'(x_i)) \tag{8}$$

Combining (6), (7), (8) and $D_1(i) = c_i$ / sum c_j.

$$\begin{aligned} \sum c_i [\![H(x_i) \neq y_i]\!] &\leq \sum c_i \cdot \exp(-y_i f'(x_i)) \\ &= \sum_i (\prod_t Z_t)(\frac{c_i}{D_1(i)}) D_{T+1}(i) \\ &= d \prod_{t=1}^{T} Z_t,\ d = \sum c_j \end{aligned} \tag{9}$$

Proof

- Requiring β(neg) ≥ β(pos) removes the cost adjustment function and the true Y label terms from H'(x)

- The Lemma also shows that H(x) is at least as accurate as H'(x)

- Clearly the upper bound is dependent on Z_t

Choosing a Good Alpha

- Estimation method (Freund and Schapire)

  Takes advantage of the case where the weak hypothesis has a range [-1,+1]

- Numerical method (Schapire and Singer)

  Differentiate Z and solve for α

Estimating Alpha

$$\begin{aligned} Z &= \sum_i D(i) e^{-\alpha u_i} \\ &\le \sum_i D(i) \left(\frac{1+u_i}{2} e^{-\alpha} + \frac{1-u_i}{2} e^{\alpha} \right) . \end{aligned} \tag{10}$$

Where $U_i = y_i h_t(x_i) \beta_i$

Minimizing the right hand side by zeroing the first derivative yields:

$$\alpha = \tfrac{1}{2} \ln \left(\frac{1+r}{1-r} \right) \tag{11}$$

Choosing the best Alpha

$$Z'(\alpha) = \frac{dZ}{d\alpha} = -\sum_i D(i) u_i e^{-\alpha u_i} = 0 \qquad (12)$$

- Since $Z''(\alpha) > 0$, $Z'(\alpha)$ can have only one zero crossing

- Authors use estimator to find candidate and use numerical method to refine

Four key questions

•Which method achieved lowest misclassification costs and how often was AdaCost the lowest?

•Quantitatively, how did the misclassification cost between AdaCost, AdaBoost, and baseline “weak” learner differ?

•How does AdaCost compare to AdaBoost round by round in terms of misclassification cost?

•Does AdaCost consume more computing power than AdaBoost?

Four key questions

- AdaCost achieved lowest misclassification costs 88% of time
- The absolute reduction in misclassification costs ranged from 0.1% to 57%
- Round by round, AdaCost has a lower misclassification cost than AdaBoost in majority of cases
- AdaCost consumes time on the same order of magnitude as AdaBoost

Data Sets

Table 1: Data Set Summary

$\mathcal{S}$	Data	Data Size	Testing Size	Positive%
1	hypothyroid	3163	CV	4.77
2	boolean	32768	CV	13.34
3	dis	2800	972	4.63
4	crx	690	CV	44.5
5	breast cancer	699	CV	34.5
6	wpbc	198	CV	23.74
7	chase	40K*10	40K*10	≈ 20

Misclassification Costs

- Credit card data:

  - Misclassification costs either (Charge – Overhead) OR (Overhead)

  - Overhead charge varied from $60 - $90

- Other sets:

  - Used set of fixed ratios ranging from 2 to 9

Training and Testing

- Credit card: One month’s worth of data used for training and data from two month’s later used for testing

- Other sets: 10-fold cross validation to average results or used specific testing data provided

Weak learner: cRipper

•Cost-sensitive modification of Ripper [Cohen]

• Provides easy way of changing the distribution in the data set

• Supplied with distribution that is linear in the cost of each instance (D1(i))

•Confidence (|h(x)|) estimated using Laplace estimate to avoid over-estimating accuracy by using training data

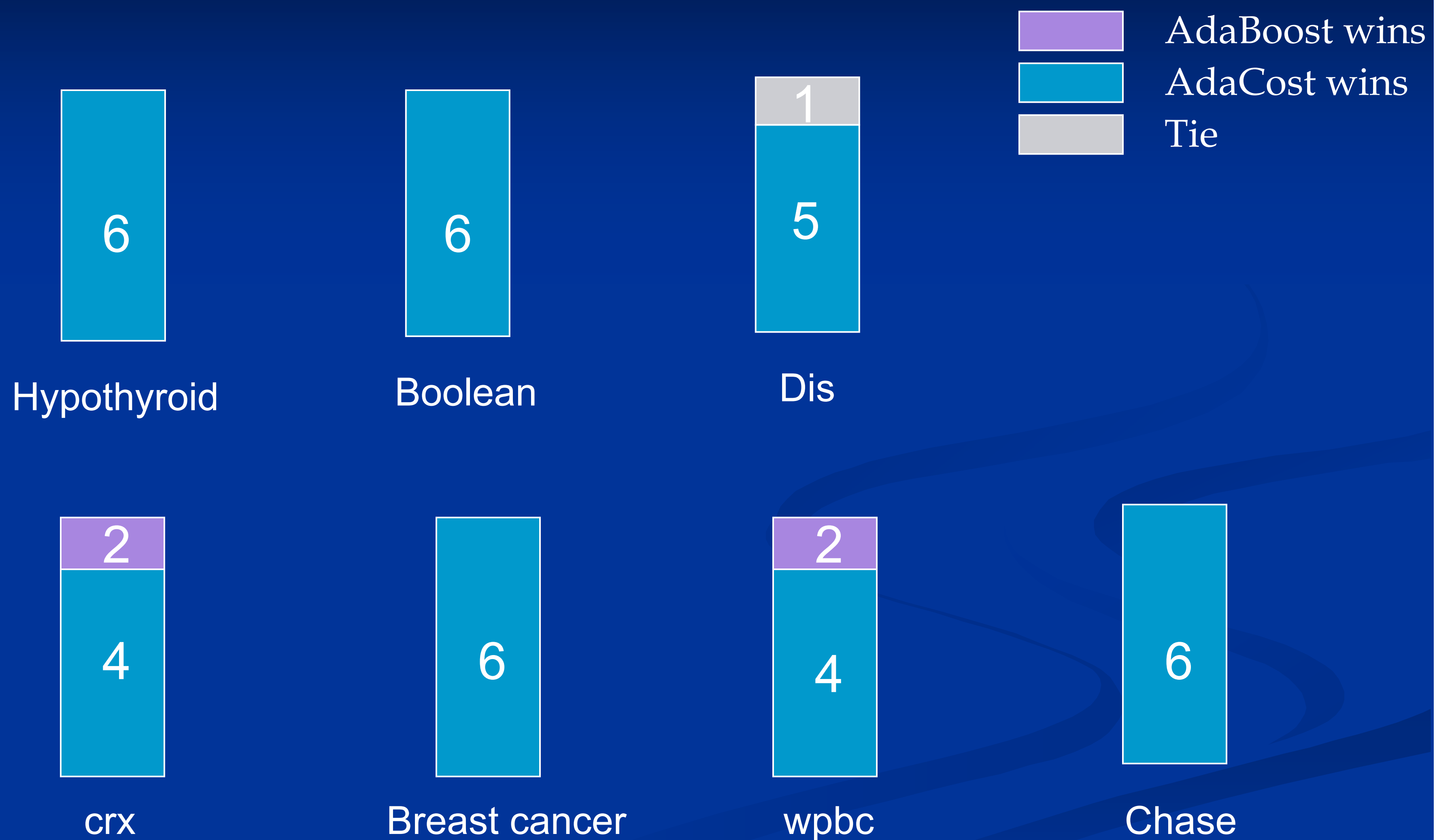
Results

Table 2: Percentage Cumulative Loss by cRIPPER, AdaBoost and AdaCost for Six Data Sets

S	R	**cRpr**	**Bst**	**Cst**	**(C-B)(%)**	**(C-P)(%)**
	2	1.4	1.6	1.2	**-0.4(-25)**	**-0.2(-16)**
	3	1.8	2.0	1.6	**-0.3(-17)**	**-0.2(-10)**
	4	2.1	2.2	1.8	**-0.4(-16)**	**-0.3(-12)**
1	5	2.5	2.7	2.2	**-0.4(-16)**	**-0.3(-11)**
	6	3.2	3.0	2.5	**-0.6(-19)**	**-0.7(-23)**
	7	3.1	2.8	2.7	**-0.1(-3)**	**-0.4(-13)**
	8	3.0	3.1	2.7	**-0.4(-12)**	**-0.3(-10)**
	9	3.0	3.2	2.5	**-0.7(-23)**	**-0.5(-17)**
	μ	2.5	2.6	2.2	**-0.4 (-16.0)**	**-0.4(-14.2)**
	2	13.8	10.5	3.3	**-7.2(-69)**	**-10.6(-76)**
	3	14.2	11.6	5.0	**-6.6(-57)**	**-9.2(-65)**
	4	15.4	10.9	6.9	**-4.0(-37)**	**-8.5(-55)**
2	5	14.7	11.4	7.3	**-4.1(-36)**	**-7.4(-50)**
	6	13.9	9.3	8.1	**-1.3(-13)**	**-5.8(-42)**
	7	19.5	9.6	8.5	**-1.1(-11)**	**-11.0(-57)**
	8	18.0	9.6	8.3	**-1.3(-14)**	**-9.6(-54)**
	9	18.3	11.0	8.1	**-3.0(-27)**	**-10.2(-56)**
	μ	16.0	10.5	6.9	**-3.6 (-34.1)**	**-9.1(-56.7)**
	2	2.3	2.6	2.0	**-0.6(-24)**	**-0.3(-14)**
	3	4.1	3.5	3.1	**-0.4(-12)**	**-1.0(-25)**
	4	5.0	4.3	4.3	0.0(0)	**-0.7(-14)**
3	5	6.2	4.9	4.4	**-0.5(-10)**	**-1.8(-29)**
	6	6.5	7.0	5.5	**-1.5(-21)**	**-1.0(-15)**
	7	7.6	8.0	6.7	**-1.2(-15)**	**-0.9(-11)**
	8	6.7	7.6	6.1	**-1.5(-20)**	**-0.6(-9)**
	9	7.8	10.1	7.1	**-3.0(-30)**	**-0.7(-9)**
	μ	5.8	6.0	4.9	**-1.1 (-18.2)**	**-0.9(-14.9)**

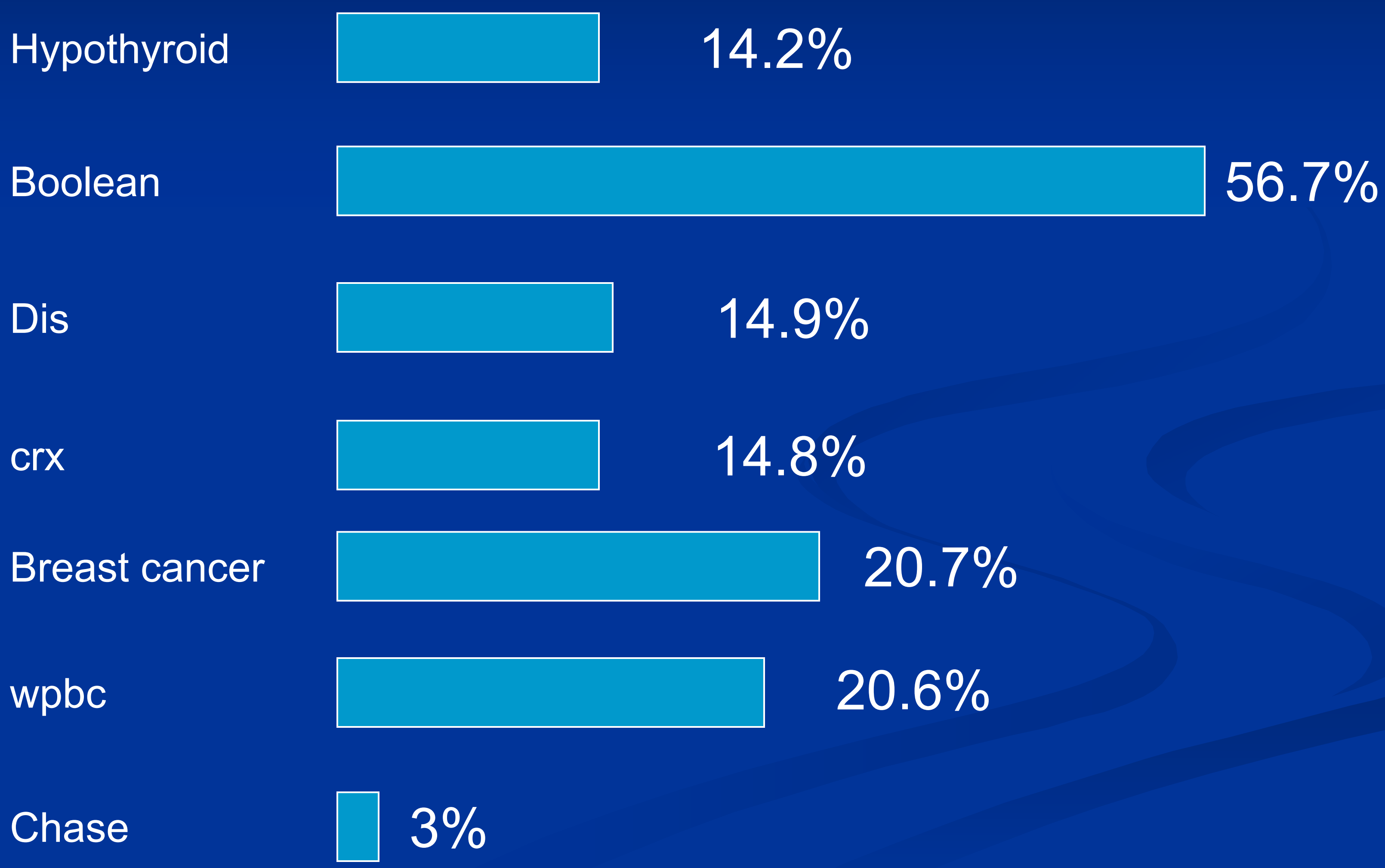
S	R	**cRpr**	**Bst**	**Cst**	**(C-B)(%)**	**(C-P)(%)**
	2	14.0	10.5	5.4	**-5.1(-48)**	**-8.6(-61)**
	3	11.7	12.4	12.1	**-0.3(-3)**	0.4(4)
	4	11.1	11.2	11.3	0.1(1)	0.2(2)
4	5	9.7	9.9	10.0	0.1(1)	0.3(3)
	6	9.8	7.4	7.3	**-0.1(-2)**	**-2.5(-25)**
	7	8.5	8.1	4.9	**-3.2(-39)**	**-3.6(-42)**
	8	8.1	10.6	8.7	**-2.0(-19)**	0.6(7)
	9	7.7	11.1	8.9	**-2.2(-20)**	1.2(15)
	μ	10.1	10.2	8.6	**-1.6 (-15.5)**	**-1.5(-14.8)**
	2	4.4	3.2	1.7	**-1.4(-45)**	**-2.7(-61)**
	3	3.7	3.4	1.8	**-1.6(-46)**	**-1.9(-51)**
	4	3.8	4.8	3.1	**-1.8(-37)**	**-0.7(-19)**
5	5	4.1	4.6	4.2	**-0.4(-9)**	0.1(2)
	6	3.5	3.6	2.2	**-1.4(-39)**	**-1.3(-38)**
	7	3.5	3.2	3.2	**-0.1(-2)**	**-0.3(-9)**
	8	3.3	3.5	2.2	**-1.4(-38)**	**-1.1(-34)**
	9	3.2	3.1	3.0	**-0.1(-3)**	**-0.2(-7)**
	μ	3.7	3.7	2.7	**-1.0 (-27.6)**	**-1.0(-27.7)**
	2	35.8	43.7	34.0	**-9.7(-22)**	**-1.8(-5)**
	3	38.9	35.1	20.5	**-14.6(-42)**	**-18.4(-47)**
	4	36.7	33.5	35.7	2.1(6)	**-1.0(-3)**
6	5	35.0	34.5	22.6	**-12.0(-35)**	**-12.4(-35)**
	6	31.2	18.7	11.1	**-7.6(-41)**	**-20.1(-64)**
	7	28.6	28.6	28.8	0.1(1)	0.2(1)
	8	24.6	24.8	25.1	0.4(2)	0.5(2)
	9	25.5	27.1	25.7	**-1.4(-5)**	0.2(1)
	μ	32.0	30.8	25.4	**-5.3 (-17.3)**	**-6.6(-20.6)**

AdaCost "wins"

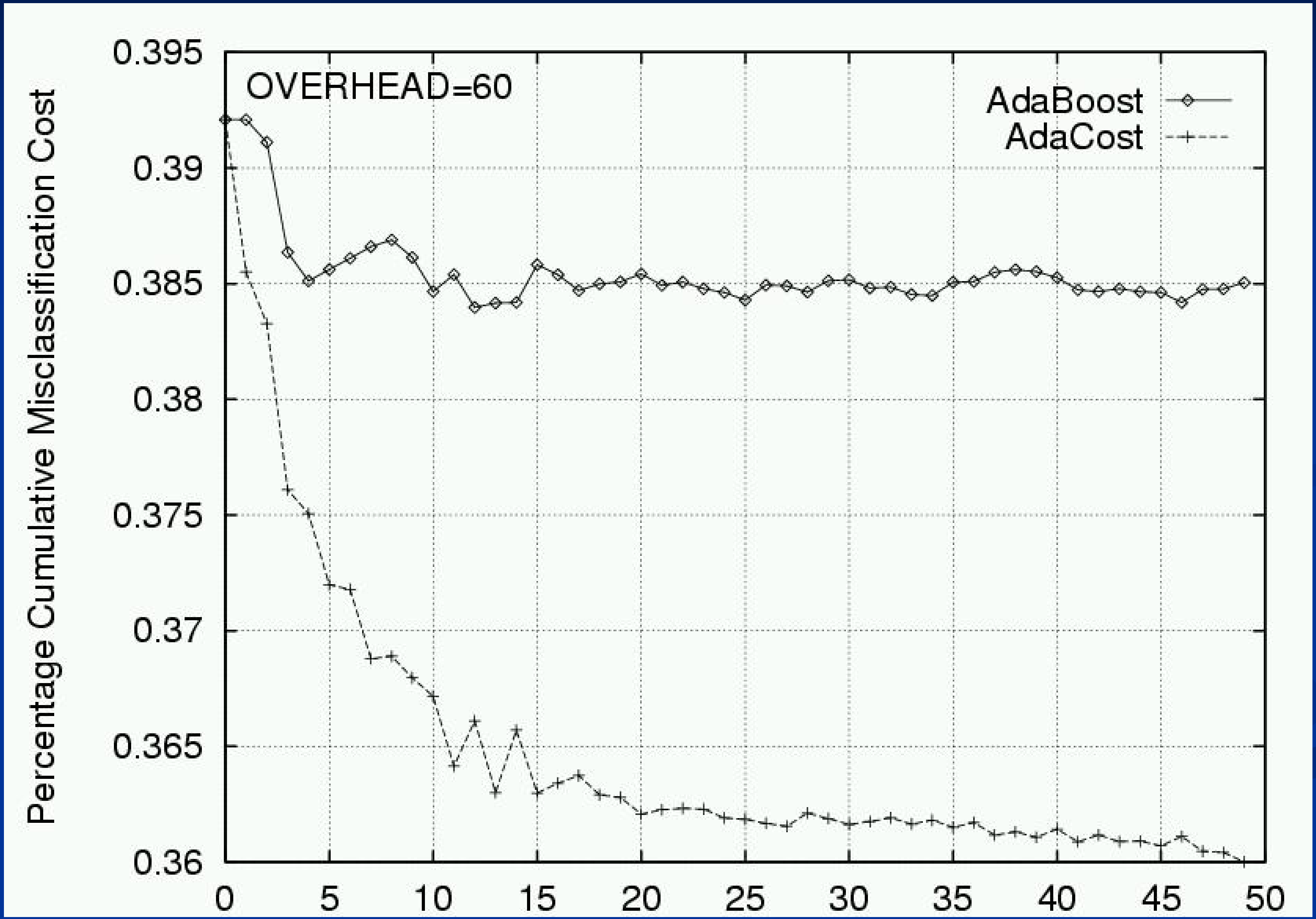


% Cumulative Loss

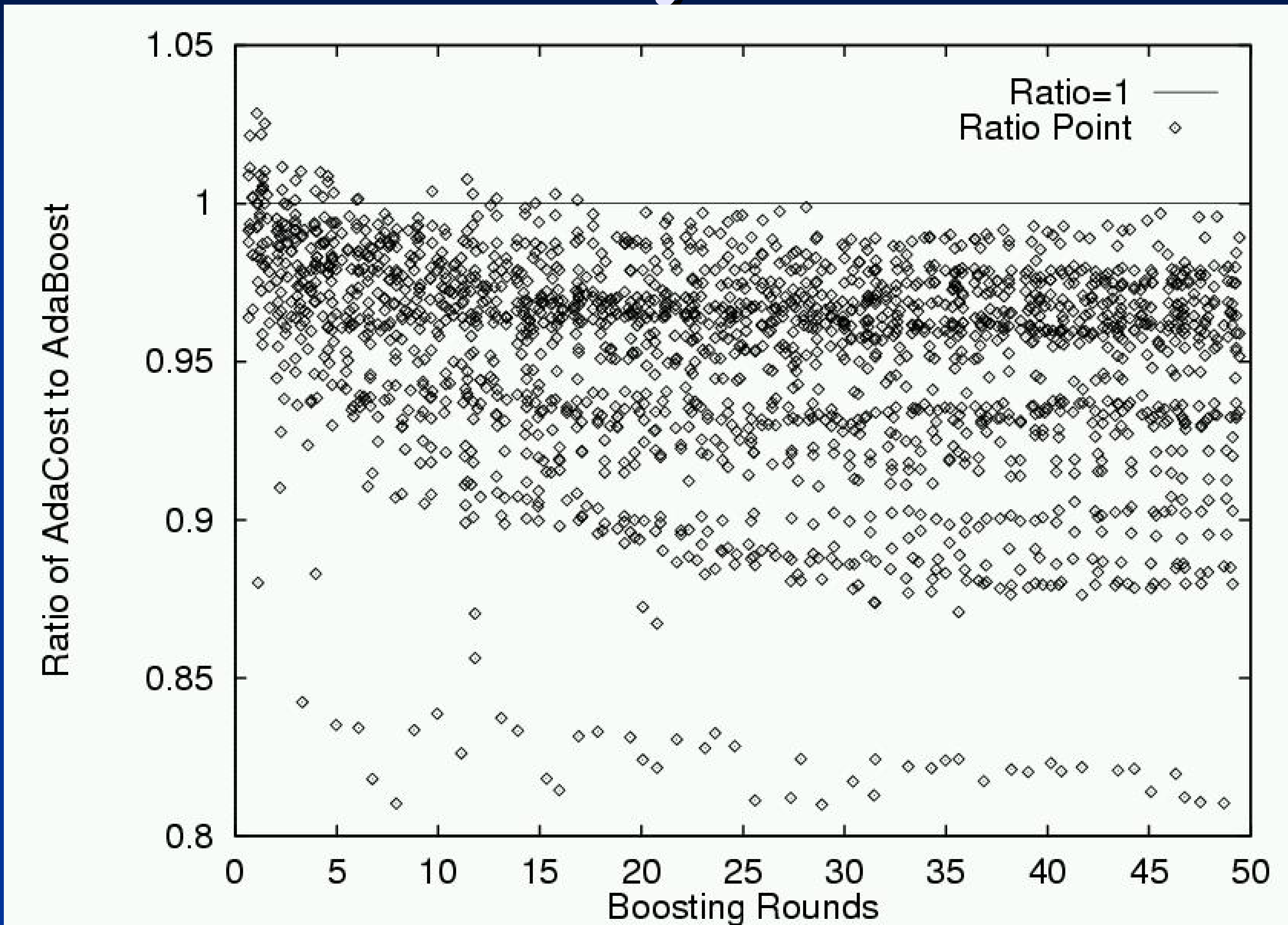
Average Improvement Over AdaBoost



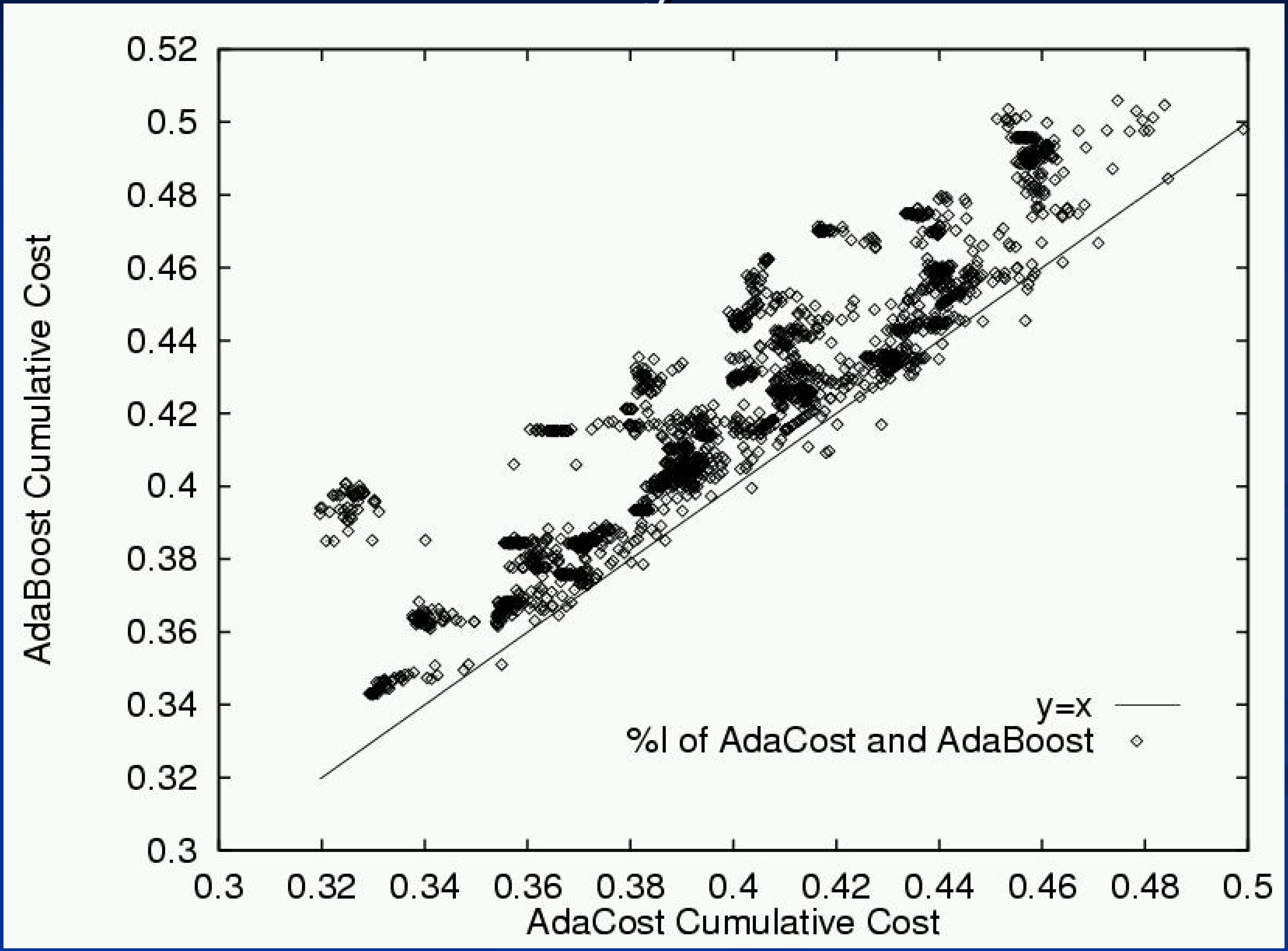
Round by Round



Round by Round



Round by Round



Time consumption

- As seen by the algorithm

- Authors present no empirical evidence comparing time consumption between AdaCost and AdaBoost

Questions on Research

- Why did AdaCost not dominate AdaBoost in all sample data sets?
- Is it necessary to reapply the cost function after the first round?
- How would results have changed with greater cost ratios ?

Other variations

- Calculating expected misclassification cost for every class and choosing class with lowest cost [Ting and Zheng]

- Giving positives a higher fixed weight than negatives [Shawe-Taylor]

Future Work

- How sensitive are methods to cost (what if cost is only an estimation)?

- How well do cost-sensitive boosting methods do towards achieving maximum misclassification cost reduction?

- Is it necessary to reapply the cost function after the first round?