

# MULTICLASS BOOSTING WITH REPARTITIONING

Ling Li

Learning Systems Group, Caltech

ICML 2006



# BINARY AND MULTICLASS PROBLEMS

- Binary classification problems  $\mathcal{Y} = \{-1, 1\}$
- Multiclass classification problems  $\mathcal{Y} = \{1, 2, \dots, K\}$
- A multiclass problem can be reduced to a collection of binary problems

## EXAMPLES

- one-vs-one
- one-vs-all
- Usually we obtain an ensemble of binary classifiers

## A UNIFIED APPROACH [ALLWEIN ET AL., 2000]

- Given a coding matrix

$$\mathbf{M} = \begin{pmatrix} - & - \\ - & + \\ + & - \\ + & + \end{pmatrix}$$

- Each row is a codeword for a class

the codeword for class 2 is “-+”

- Construct a binary classifier for each column (partition)

$f_1$  should discriminate classes 1 and 2 from 3 and 4

- Decode  $(f_1(\mathbf{x}), f_2(\mathbf{x}))$  to predict

$(f_1(\mathbf{x}), f_2(\mathbf{x})) = (+, +)$  predicts class label 4

# CODING MATRIX

## ERROR-CORRECTING

- If a few binary classifiers make mistakes, the correct label can still be predicted
- Assure the Hamming distance between codewords is large

$$\begin{pmatrix} - & - & - & + & + \\ - & + & + & - & + \\ + & - & + & - & - \\ + & + & - & + & - \end{pmatrix}$$

- Assume errors are independent

## EXTENSIONS

- Some entries can be 0
- Various distance measures can be used

# MULTICLASS BOOSTING [GURUSWAMI & SAHAI, 1999]

## PROBLEMS

- Errors of the binary classifiers may be highly correlated
- Optimal coding matrix is problem dependent

## BOOSTING APPROACH

- Dynamically generates the coding matrix
- Reweights examples to reduce the error correlation
- Minimizes a multiclass margin cost

## PROTOTYPE

- The ensemble  $\mathbf{F} = (f_1, f_2, \dots, f_T)$
- $f_t$  has a coefficient  $\alpha_t$
- The Hamming distance

$$\Delta(\mathbf{M}(k), \mathbf{F}(\mathbf{x})) = \sum_{t=1}^T \alpha_t \frac{1 - \mathbf{M}(k, t) f_t(\mathbf{x})}{2}.$$

## MULTICLASS BOOSTING

- 1:  $\mathbf{F} \leftarrow (0, 0, \dots, 0)$ , i.e.,  $f_t \leftarrow 0$
- 2: **for**  $t = 1$  to  $T$  **do**
- 3:   Pick the  $t$ -th column  $\mathbf{M}(\cdot, t) \in \{-, +\}^K$
- 4:   Train a binary hypothesis  $f_t$  on  $\{(\mathbf{x}_n, \mathbf{M}(y_n, t))\}_{n=1}^N$
- 5:   Decide a coefficient  $\alpha_t$
- 6: **end for**
- 7: **return**  $\mathbf{M}$ ,  $\mathbf{F}$ , and  $\alpha_t$ 's

# MULTICLASS MARGIN COST

For an example  $(\mathbf{x}, y)$ , we want

$$\Delta(\mathbf{M}(k), \mathbf{F}(\mathbf{x})) > \Delta(\mathbf{M}(y), \mathbf{F}(\mathbf{x})), \quad \forall k \neq y$$

## MARGIN

The margin of the example  $(\mathbf{x}, y)$  for class  $k$  is

$$\rho_k(\mathbf{x}, y) = \Delta(\mathbf{M}(k), \mathbf{F}(\mathbf{x})) - \Delta(\mathbf{M}(y), \mathbf{F}(\mathbf{x}))$$

## EXPONENTIAL MARGIN COST

$$C(\mathbf{F}) = \sum_{n=1}^N \sum_{k \neq y_n} e^{-\rho_k(\mathbf{x}_n, y_n)}$$

This is similar to the binary exponential margin cost.

## GRADIENT DESCENT [SUN ET AL., 2005]

A multiclass boosting algorithm can be deduced as gradient descent on the margin cost

## MULTICLASS BOOSTING

- 1:  $\mathbf{F} \leftarrow (0, 0, \dots, 0)$ , i.e.,  $f_t \leftarrow 0$
- 2: **for**  $t = 1$  to  $T$  **do**
- 3:   Pick  $\mathbf{M}(\cdot, t)$  and  $f_t$  to maximize the negative gradient
- 4:   Pick  $\alpha_t$  to minimize the cost along the gradient
- 5: **end for**
- 6: **return**  $\mathbf{M}$ ,  $\mathbf{F}$ , and  $\alpha_t$ 's

AdaBoost.ECC is a concrete algorithm on the exponential cost.



## GRADIENT OF EXPONENTIAL COST

skipped most math equations

Say  $\mathbf{F} = (f_1, \dots, f_t, 0, \dots)$ .

$$-\left. \frac{\partial C(\mathbf{F})}{\partial \alpha_t} \right|_{\alpha_t=0} = U_t(1 - 2\varepsilon_t)$$

- $\tilde{D}_t(n, k) = e^{-\rho_k(\mathbf{x}_n, y_n)}$  (before  $f_t$  is added)  
How would this example of class  $y_n$  be confused as class  $k$ ?
- $U_t = \sum_{n=1}^N \sum_{k=1}^K \tilde{D}_t(n, k) \llbracket \mathbf{M}(k, t) \neq \mathbf{M}(y_n, t) \rrbracket$   
Sum of the “confusion” for binary relabeled examples
- $D_t(n) = U_t^{-1} \cdot \sum_{k=1}^K \tilde{D}_t(n, k) \llbracket \mathbf{M}(k, t) \neq \mathbf{M}(y_n, t) \rrbracket$   
Sum of the “confusion” for individual example
- $\varepsilon_t = \sum_{n=1}^N D_t(n) \llbracket f_t(\mathbf{x}_n) \neq \mathbf{M}(y_n, t) \rrbracket$

# PICKING PARTITIONS

$$-\left. \frac{\partial C(\mathbf{F})}{\partial \alpha_t} \right|_{\alpha_t=0} = U_t(1 - 2\varepsilon_t)$$

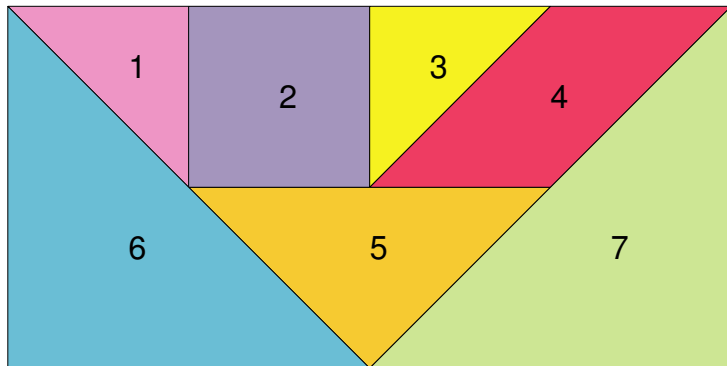
- $U_t$  is determined by the  $t$ -th column/partition
- $\varepsilon_t$  is also decided by the binary learning performance
- Seems that we should pick the partition to maximize  $U_t$  and ask the binary learner to minimize  $\varepsilon_t$

## PICKING PARTITIONS

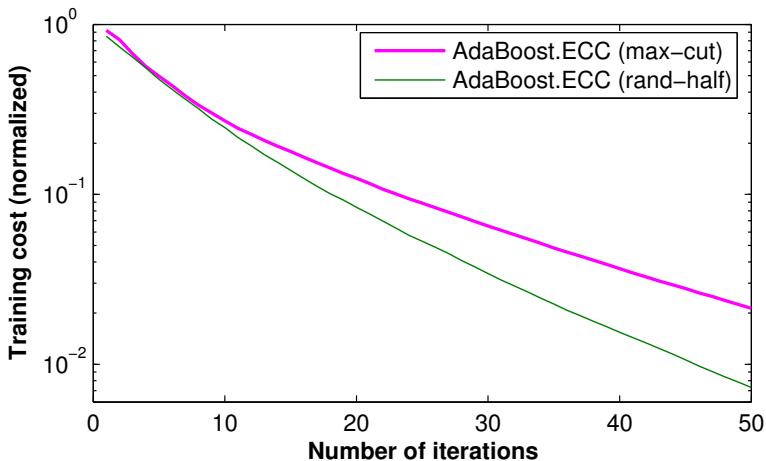
- max-cut: picks the partition with the largest  $U_t$
- rand-half: randomly assigns + to half of the classes

Which one would you pick?

# TANGRAM

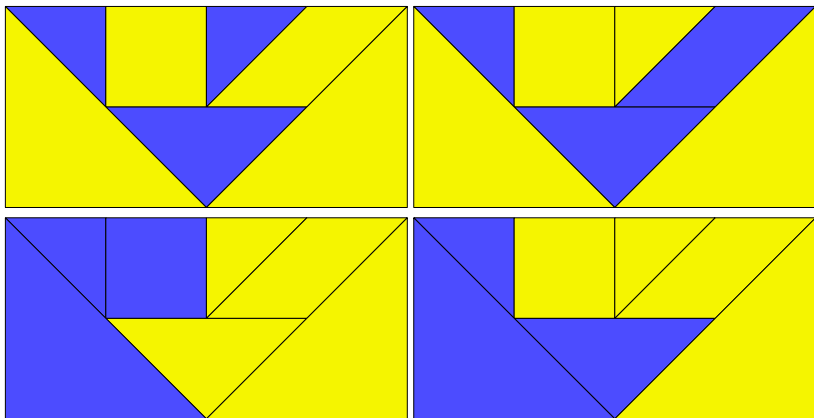


## MARGIN COST (WITH PERCEPTRONS)



# WHY WAS MAX-CUT WORSE?

- Maximizing  $U_t$  brings strong error-correcting ability
- But it also generates much “hard” binary problems



## TRADE-OFF

$$-\left. \frac{\partial C(\mathbf{F})}{\partial \alpha_t} \right|_{\alpha_t=0} = U_t(1 - 2\varepsilon_t)$$

- Hard problems deteriorate the binary learning, thus overall the negative gradient might be smaller
- Need to find a trade-off between  $U_t$  and  $\varepsilon_t$
- The “hardness” depends on the binary learner
- So we may “ask” the binary learner for a better partition

# REPARTITIONING

- Given a binary classifier  $f_t$ , which partition is the best?
- The one that maximizes  $-\frac{\partial C(\mathbf{F})}{\partial \alpha_t} \Big|_{\alpha_t=0}$

skipped most math equations

$\mathbf{M}(k, t)$  can be decided from the output of  $f_t$  and the “confusion”

# ADABOOST.ERP

- Given a partition, a binary classifier can be learned
- Given a binary classifier, a better partition can be generated
- These two steps can be carried out alternatively
- We use a string of “L” and “R” to denote the schedule

## EXAMPLE

“LRL” means “Learning  $\rightarrow$  Repartitioning  $\rightarrow$  Learning”

- We can also start from partial partitions

## EXAMPLE

rand-2 starts with two random classes

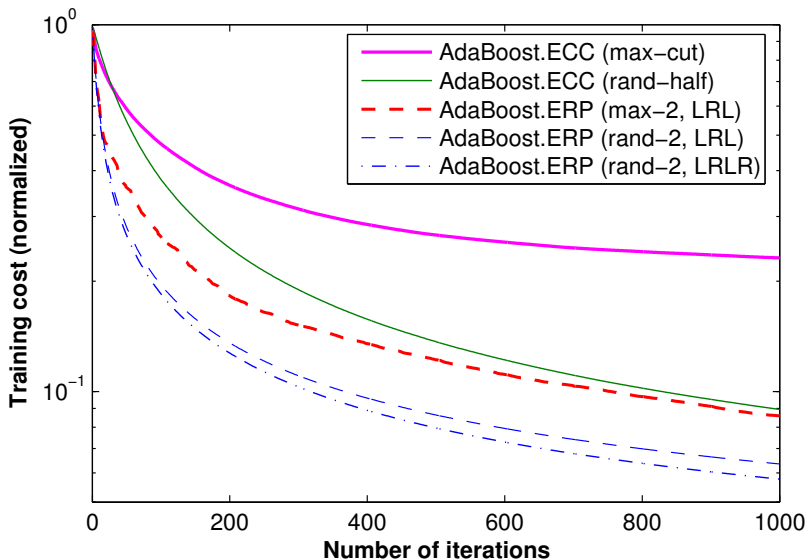
Faster learning; focus on local class structure



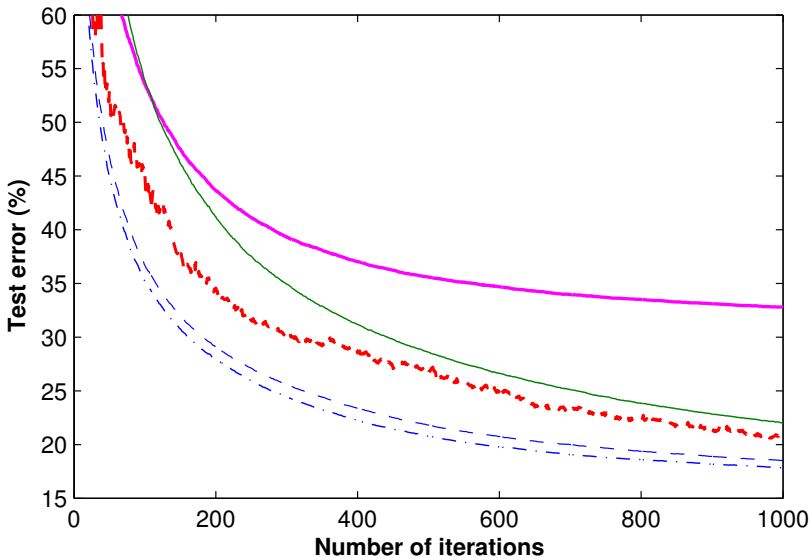
# EXPERIMENT SETTINGS

- We compared one-vs-one, one-vs-all, AdaBoost.ECC, and AdaBoost.ERP
- Four different binary learners: decision stumps, perceptrons, binary AdaBoost, and SVM-perceptron
- Ten UCI data sets with number of classes varies from 3 to 26

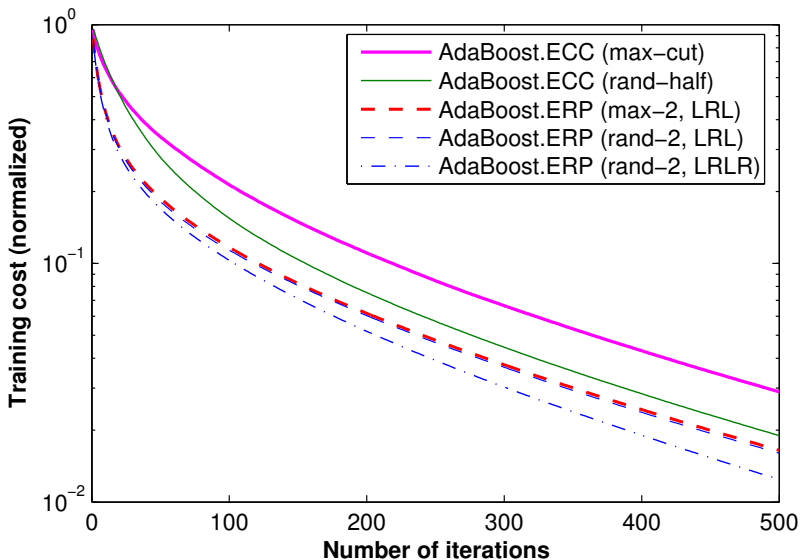
## COST WITH DECISION STUMPS ON LETTER



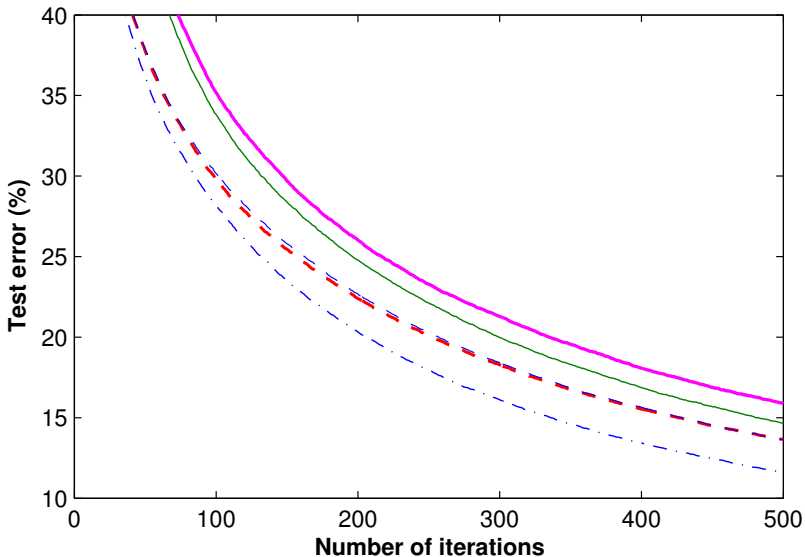
## TEST ERROR WITH DECISION STUMPS ON LETTER



## COST WITH PERCEPTRONS ON LETTER



## TEST ERROR WITH PERCEPTRONS ON LETTER



# OVERALL RESULTS

- AdaBoost.ERP achieved the lowest cost, and the lowest test error on most of the data sets
- The improvement is especially significant for weak binary learners
- With SVM-perceptron, all methods were comparable
- AdaBoost.ERP starting with partial partitions were much faster than AdaBoost.ECC
- One-vs-one is much worse with weak binary learners
- One-vs-one is much faster

# SUMMARY

- A multiclass problem can be reduced to a collection of binary problems via an error-correcting coding matrix
- Multiclass boosting dynamically generates the coding matrix and the binary problems
- Hard binary problems deteriorate the binary learning
- AdaBoost.ERP achieves a better trade-off between the error-correcting and the binary learning