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 $\mathcal{Y} = \{1, 2, \ldots, K\}$
- 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 $\mathbf{F} = (f_1, f_2, \ldots, f_T)$
- f_t has a coefficient α_t
- The Hamming distance

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

Multiclass Boosting

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

Multiclass Margin Cost

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

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

Margin

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

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

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: $\mathbf{F} \leftarrow (0, 0, \ldots, 0)$, 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 α_t to minimize the cost along the gradient
5: **end for**
6: **return** $\mathbf{M}$, $\mathbf{F}$, and α_t's

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

Gradient of Exponential Cost

skipped most math equations

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

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

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

Picking Partitions

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

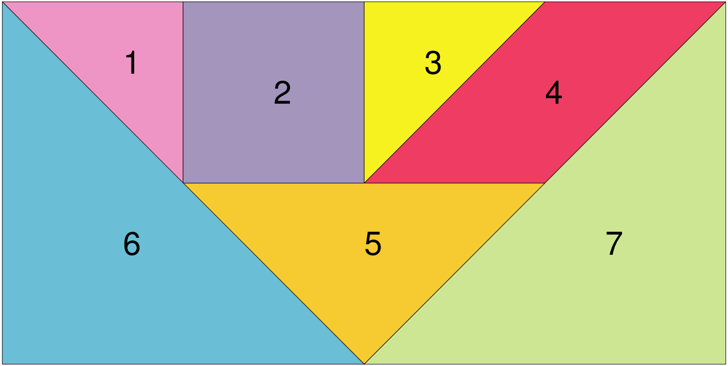
- U_t is determined by the t-th column/partition
- ε_t is also decided by the binary learning performance
- Seems that we should pick the partition to maximize U_t and ask the binary learner to minimize ε_t

Picking Partitions

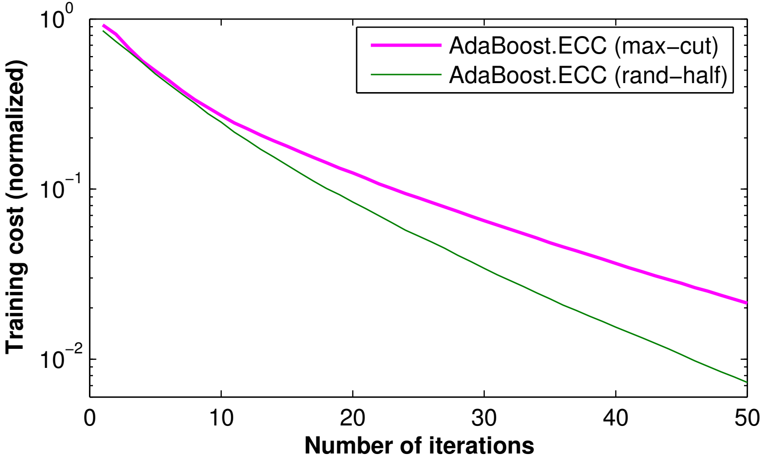
- max-cut: picks the partition with the largest U_t
- rand-half: randomly assigns $+$ to half of the classes

Which one would you pick?

Tangram

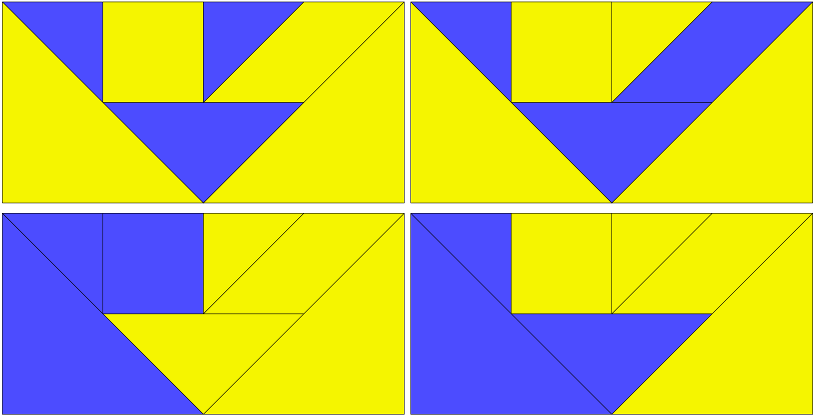


Margin Cost (with perceptrons)



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 ε_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 $-\left.\dfrac{\partial C(\mathbf{F})}{\partial \alpha_t}\right|_{\alpha_t=0}$

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 → Repartitioning → Learning”

- We can also start from partial partitions

Example

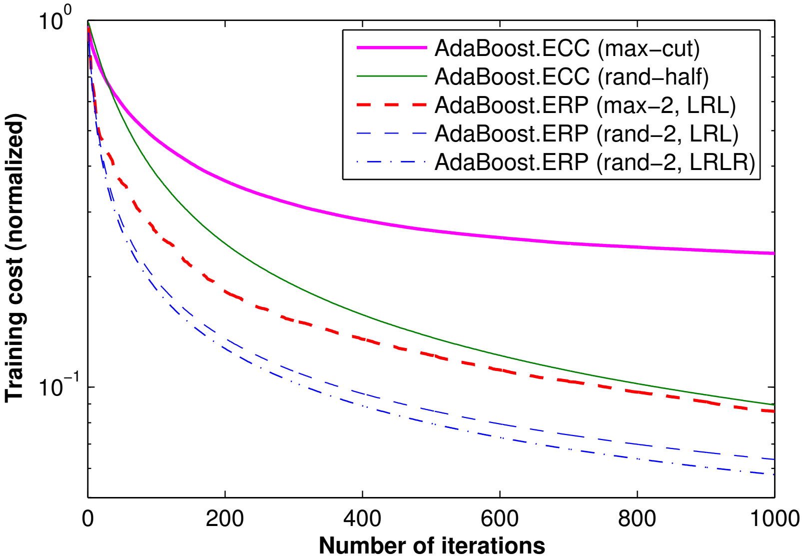
rand-2 starts with two random classes

Faster learning; focus on local class structure

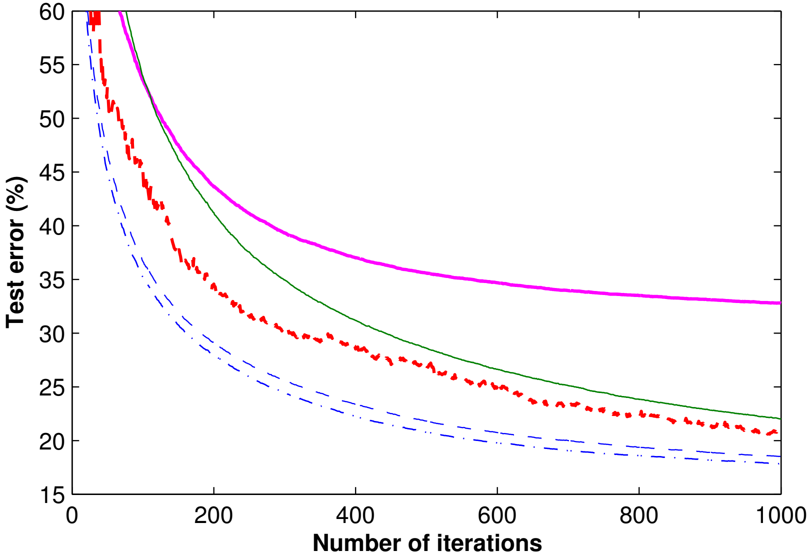
Experiment Settings

- 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

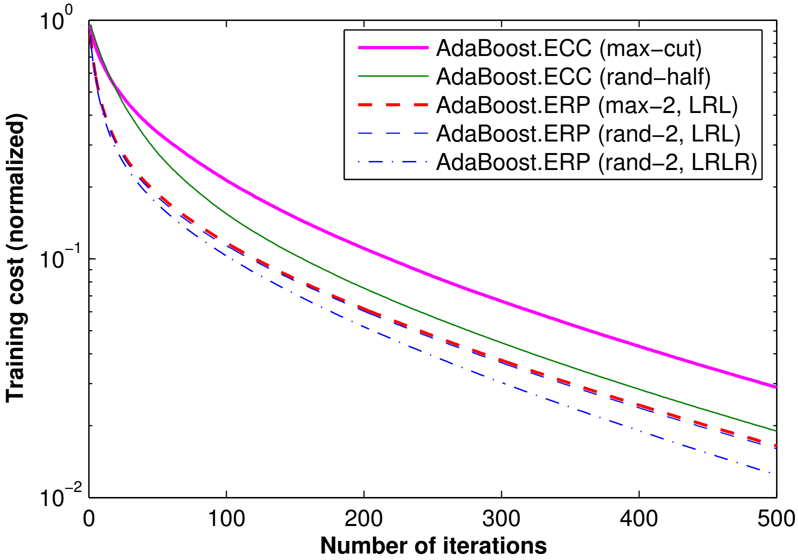
Cost with Decision Stumps on letter



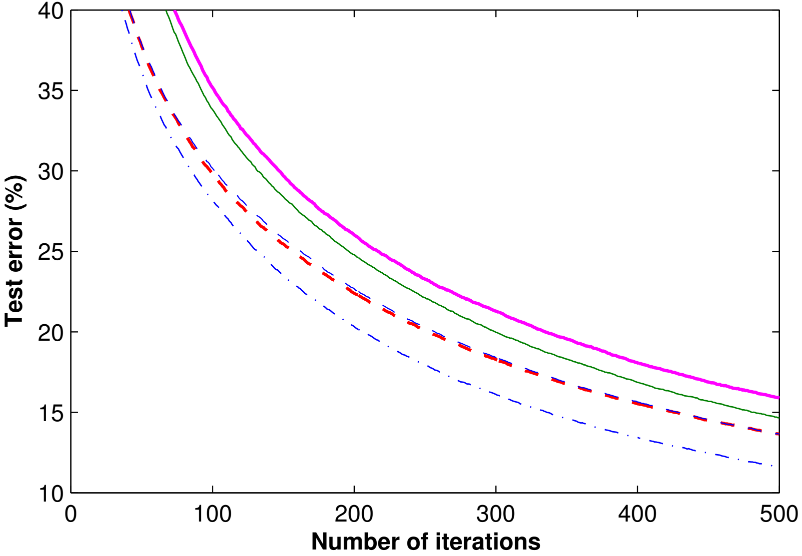
Test Error with Decision Stumps on letter



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