Multiclass Boosting with Repartitioning

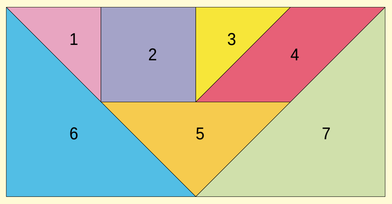
Ling Li, Learning Systems Group, Caltech

I

Multiclass Classification

Considering a classification problem where the set of labels is $\mathcal{Y}$.

- Binary classification: $\mathcal{Y} = \{-1, +1\}$
- Multiclass classification: $\mathcal{Y} = \{1, \ldots, K\}$



A multiclass problem can be reduced to a collection of binary problems via approaches such as one-vs-one and one-vs-all. Most of the approaches can be unified with an error-correcting coding matrix.[1]

An example coding matrix $\mathbf{M} = \begin{pmatrix} - & - & - & + \\ - & + & + & - \\ + & - & + & - \\ + & + & - & - \end{pmatrix}$.

- Each row is a codeword for a class.
- Each column is some partition of the class labels.
- A binary classifier f_t is constructed for the t-th partition.
- Decode $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_T(\mathbf{x}))$ to predict y, say, with the shortest weighted Hamming distance

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

II

Multiclass Boosting

The optimal coding matrix is problem-dependent. A boosting-style algorithm can dynamically generate a coding matrix:

- start from an empty coding matrix,
- iteratively pick partitions and learn binary classifiers, and
- minimize a multiclass margin cost $C(\mathbf{F})$.

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

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

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

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

A specific boosting algorithm, AdaBoost.ECC[2], optimizes

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

A multiclass boosting algorithm can be deduced as gradient descent on the margin cost.[3] To determine the t-th binary classifier f_t, we try to maximize the negative gradient at $\alpha_t = 0$,

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

- U_t is determined by the t-th partition, and reflects the error-correcting ability added by the t-th column;
- ε_t is the training error of the binary classifier f_t.

It seems that we should maximize U_t and minimize ε_t.

III

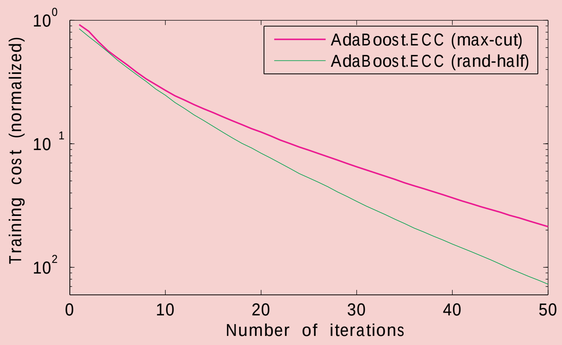
Picking Partitions

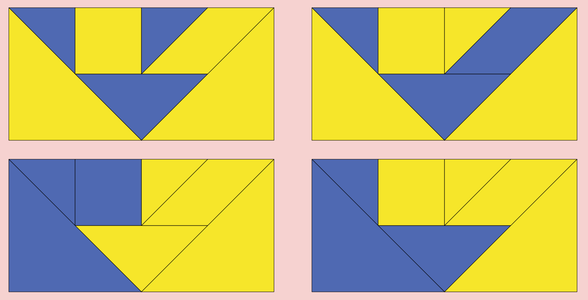
For the tangram problem (left), we use straight lines (perceptrons) as the binary classifiers with the following two partition-picking methods:

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

The random method rand-half actually works better.

- Maximizing U_t brings strong error-correcting ability, but
- it also generates much "hard" binary problems.





↰ Dominating partitions in the tangram experiment: (above) with max-cut; (below) with rand-half

IV

Boosting with Repartitioning

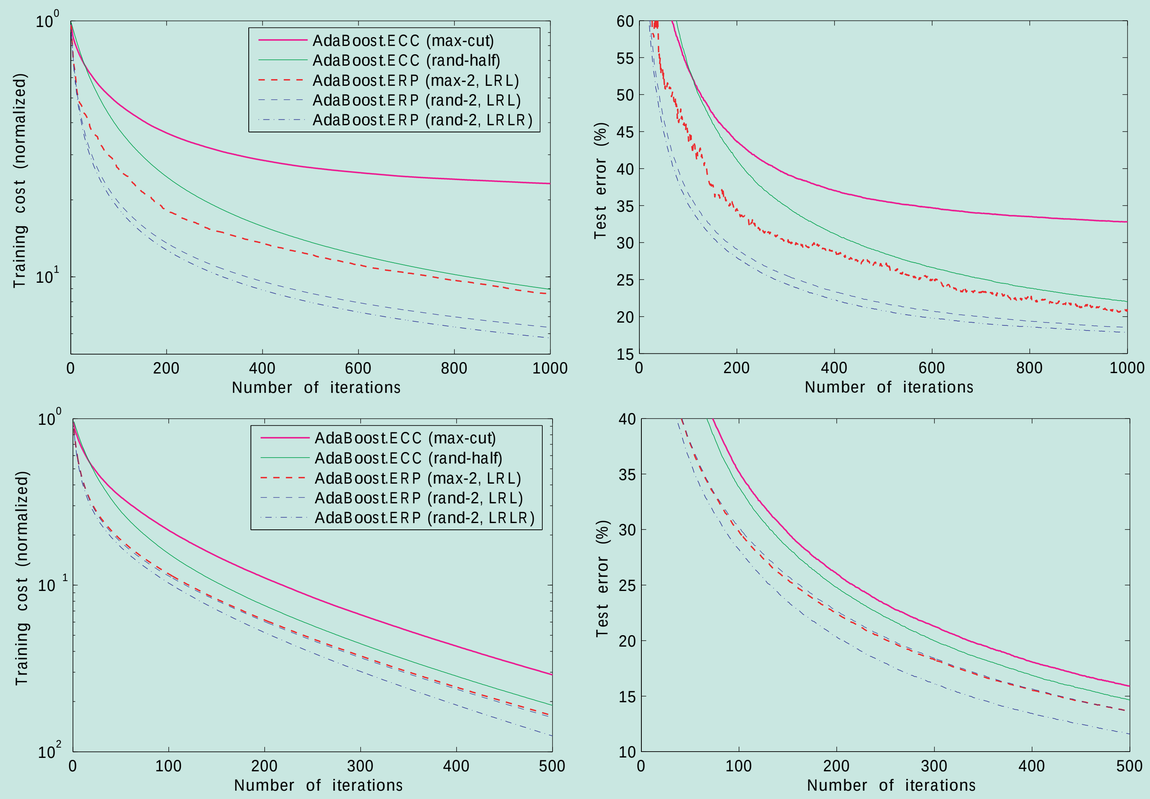
Hard problems deteriorate the binary learning as well as the negative gradient. We need to find a better trade-off between U_t and ε_t.

- The "hardness" depends on the partition and the binary learner.
- We may "ask" the binary learner for a better partition.

Repartitioning is designed to generate a better partition from the current binary classifier.

- "L": Given a partition, a binary classifier can be learned.
- "R": Given a binary classifier, a better partition can be generated.
- These two steps can be carried out alternatively.

AdaBoost.ERP, which is AdaBoost.ECC with repartitioning, finds better partitions and achieves much lower margin cost and test error.



Multiclass boosting on the letter data set, with the decision stump (above) and the perceptron (below) as the binary learner

[1] Allwein, Schapire, and Singer, 2000

[2] Guruswami and Sahai, 1999

[3] Sun, Todorovic, Li, and Wu, 2005