Randomness and Computation
Joint Workshop "New Horizons in Computing" and "Statistical Mechanical Approach to Probabilistic Information Processing" (18-21 July, 2005, Sendai, Japan)

Stochastic Filtering for On-line Boosting

Noboru Murata [1]
School of Science and Engineering, Waseda University,
Tokyo 169-8555, Japan

Boosting algorithms have been successfully applied many practical classification problems. In boosting algorithms, such as AdaBoost [3], a "strong" classifier is constructed by means of a majority vote of "weak" classifiers, which are sequentially trained by intentionally changing the importance of each example with filtering, re-weighting or re-sampling [2, 3, 4, 6]. The basic idea of boosting is collecting a rich variety of classifiers and compensating weak parts of each of them with the others by voting.

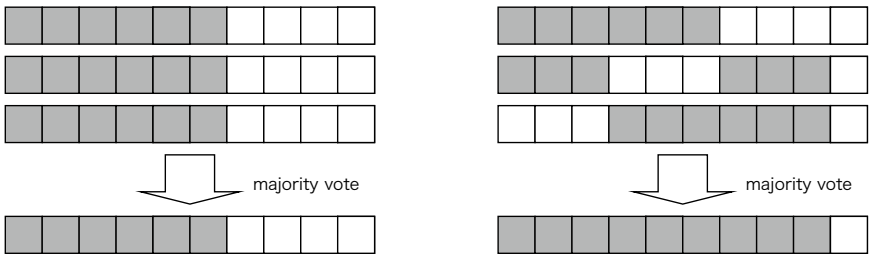

Figure 1: Simple explanation of boosting by majority vote.

As for a simple example, let us consider a vote of three classifiers. Assume that each classifier is "weak" and can correctly discriminate only 60% of the data. If the classifiers have poor variety, a majority vote does not effectively work and it can correctly classify about 60% of the input data as shown in Fig.1 left. On the other hand, if their variety is rich enough, the voting classifier works quite effectively and it can discriminate 90% as shown in Fig.1 right. In the case of three classifiers, the voting works well if two of three classifiers give correct answers for given inputs, then the rest can devote its capacity to other inputs. This is the key issue how weak classifiers are boosted to a single strong classifier.

Let us consider a classification problem of predicting a label y in a binary set $\{-1, 1\}$ for a given feature vector x which belongs to some space $\mathcal{X}$. A simple implementation of boosting three classifiers is given by [6] and summarized in Fig.2. This algorithm gives a way for constructing classifiers sequentially, but

step 1: collect examples D_1 from the target distribution P, and train h_1 with D_1.

step 2: collect examples D_2 with filter A, and train h_2 with D_2.

filter A: flip a fair coin
head: pick an example (x, y) s.t. $h_1(x) = y$.
tail: pick an example (x, y) s.t. $h_1(x) \neq y$.

step 3: collect examples D_3 with filter B, and train h_3 with D_3.

filter B: pick an example (x, y) s.t. $h_1(x) \neq h_2(x)$.

output: a majority vote of h_1, h_2, h_3

$$H(x) = \mathrm{sign}(h_1(x) + h_2(x) + h_3(x)).$$

Figure 2: Algorithm of boosting by filter.

[1] E-mail: noboru.murata@eb.waseda.ac.jp

input: three different classifiers h_1, h_2, h_3.

step 1: pick an example (x, y).

step 2: choose $i \in \{1, 2, 3\}$ randomly.

step 3: calculate $H(x) = \sum_{j \neq i} h_j(x)$.

step 4: if $H(x) = 0$, train h_i with (x, y) otherwise, flip a fair/unfair coin

head: train h_i with (x, y)

tail: train h_i with $(x, -y)$.

step 5: goto step 1 until some condition is fulfilled.

Figure 3: outline of stochastic filter

the procedure is not good for "on-line" learning where examples are given one after another. In practice, the on-line learning situation frequently appears, where the target classification rule varies slowly and the classifier has to pursue the changing rule, for example. Also some weak classifiers, such as nonlinear discriminate functions based on neural networks [1], can be trained with on-line algorithms and their performance can be gradually improved with additionally given examples.

In the case of on-line learning, the influence of past examples gradually vanishes. Old rules are forgotten and the classifier adapts to new examples recently given. Hence if all the classifiers are trained with new examples, their behaviors become similar and their variety is gradually lost. Therefore, to develop the on-line boosting algorithm, how to keep their variety is most important.

One of the methods for keeping the variety of the classifiers is a random filter of examples; one classifier is chosen randomly and trained by a new example. With a simple random filter, however, their variety can not be kept sufficiently, because each classifier learns and approaches the target rule individually. We propose a modified random filter with negative examples, in which the class label y is intentionally reversed. We discuss statistical property of this filter and also investigate its performance experimentally. Moreover, some classifiers are associated with confidence rates of decision, such as log-odds of discriminate functions or internal states of neural networks, and we also discuss a random filter of examples weighted by those confidence rates from a geometrical viewpoint of boosting algorithms [5, 7].

References

[1] C. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[2] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2): 256–285, 1995.

[3] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, aug 1997.

[4] T. Hastie, R. Tibishirani, and J. Friedman. *The elements of statistical learning*. Springer, New York, 2001.

[5] N. Murata, T. Takenouchi, T. Kanamori, and S. Eguchi. Information Geometry of U-Boost and Bregman Divergence. *Neural Computation*, 16(7):1437–1481, 2004.

[6] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[7] T. Takenouchi and S. Eguchi. Robustifying AdaBoost by adding the naive error rate. *Neural Computation*, 16(4):767–787, 2004.