JMLR: Workshop and Conference Proceedings vol 35:1–8, 2014

Open Problem: A (missing) boosting-type convergence result for AdaBoost.MH with factorized multi-class classifiers

Balázs Kégl BALAZS.KEGL@GMAIL.COM
LAL/LRI, University of Paris-Sud, CNRS, 91898 Orsay, France

Abstract

In (Kégl, 2014), we recently showed empirically that AdaBoost.MH is one of the best multi-class boosting algorithms when the classical one-against-all base classifiers, proposed in the seminal paper of Schapire and Singer (1999), are replaced by factorized base classifiers containing a binary classifier and a vote (or code) vector. In a slightly different setup, a similar factorization coupled with an iterative optimization of the two factors also proved to be an excellent approach (Gao and Koller, 2011). The main algorithmic advantage of our approach over the original setup of Schapire and Singer (1999) is that trees can be built in a straightforward way by using the binary classifier at inner nodes. In this open problem paper we take a step back to the basic setup of boosting generic multi-class factorized (Hamming) classifiers (so no trees), and state the classical problem of boosting-like convergence of the training error. Given a vote vector, training the classifier leads to a standard weighted binary classification problem. The main difficulty of proving the convergence is that, unlike in binary AdaBoost, the sum of the weights in this weighted binary classification problem is less than one, which means that the lower bound on the edge, coming from the weak learning condition, shrinks. To show the convergence, we need a (uniform) lower bound on the sum of the weights in this derived binary classification problem.

Let the training data be $\mathcal{D} = \big\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\big\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ are *observation* vectors, and $\mathbf{y}_i \in \{\pm 1\}^K$ are *label* vectors. Sometimes we will use the notion of an $n \times d$ *observation matrix* of $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and an $n \times K$ *label matrix* $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ instead of the set of pairs $\mathcal{D}$.[1] In *multi-class* classification, the single label $\ell(\mathbf{x})$ of the observation $\mathbf{x}$ comes from a finite set. Without loss of generality, we will suppose that $\ell \in \mathcal{L} = \{1, \ldots, K\}$. The label vector $\mathbf{y}$ is a *one-hot* representation of the correct class: the $\ell(\mathbf{x})$th element of $\mathbf{y}$ will be 1 and all the other elements will be -1. To avoid confusion, from now on we will call $\mathbf{y}$ and ℓ the label and the label *index* of $\mathbf{x}$, respectively.

The goal of the AdaBoost.MH algorithm (Schapire and Singer 1999; Appendix B) is to return a vector-valued discriminant function $\mathbf{f}^{(T)} : \mathbb{R}^d \to \mathbb{R}^K$ with a small Hamming loss

$$\widehat{R}_{\text{H}}(\mathbf{f}, \mathbf{W}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{\ell=1}^{K} w_{i,\ell} \mathbb{I}\left\{\text{sign}\big(f_\ell(\mathbf{x}_i)\big) \neq y_{i,\ell}\right\}, \tag{1}$$

where $\mathbf{W} = \big[w_{i,\ell}\big]$ is an $n \times K$ weight matrix over data points and labels, usually defined as

$$w_\ell = \begin{cases} \frac{1}{2} & \text{if } \ell = \ell(\mathbf{x}) \text{ (i.e., if } y_\ell = 1), \\ \frac{1}{2(K-1)} & \text{otherwise (i.e., if } y_\ell = -1). \end{cases} \tag{2}$$

[1] We will use bold capitals $\mathbf{X}$ for matrices, bold small letters $\mathbf{x}_i$ and $\mathbf{x}_{,.j}$ for its row and column vectors, respectively, and italic for its elements $x_{i,j}$.

ADABOOST.MH directly minimizes a surrogate, the *weighted multi-class exponential margin-based error*

$$\widehat{R}_{\text{EXP}}\big(\mathbf{f}^{(T)}, \mathbf{W}\big) = \frac{1}{n}\sum_{i=1}^{n}\sum_{\ell=1}^{K} w_{i,\ell} \exp\big(-f_\ell^{(T)}(\mathbf{x}_i) y_{i,\ell}\big). \tag{3}$$

Since $\exp(-\rho) \geq \mathbb{I}\{\rho < 0\}$, (3) upper bounds (1). ADABOOST.MH builds the final discriminant function $\mathbf{f}^{(T)}(\mathbf{x}) = \sum_{t=1}^{T} \mathbf{h}^{(t)}(\mathbf{x})$ as a sum of T *base classifiers* $\mathbf{h}^{(t)} : \mathbb{R}^d \to \mathbb{R}^K$ returned by a *base learner* algorithm $\text{BASE}\big(\mathbf{X}, \mathbf{Y}, \mathbf{W}^{(t)}\big)$ in each iteration t.

Since

$$\widehat{R}_{\text{EXP}}\big(\mathbf{f}^{(T)}, \mathbf{W}\big) = \prod_{t=1}^{T} Z\big(\mathbf{h}^{(t)}, \mathbf{W}^{(t)}\big),$$

where

$$Z\big(\mathbf{h}, \mathbf{W}^{(t)}\big) = \sum_{i=1}^{n}\sum_{\ell=1}^{K} w_{i,\ell}^{(t)} \exp\big(-h_{\ell_i} y_{i,\ell}\big) \tag{4}$$

the goal of the base classifier is to maximize $Z\big(\mathbf{h}, \mathbf{W}^{(t)}\big)$. Assuming *factorized* base classifiers

$$\mathbf{h}(\mathbf{x}) = \alpha \mathbf{v} \varphi(\mathbf{x}), \tag{5}$$

where $\alpha \in \mathbb{R}^+$ is a positive real valued *base coefficient*, $\mathbf{v}$ is an input-independent *vote* (or *code*) vector of length K, and $\varphi(\mathbf{x})$ is a label-independent binary (scalar) classifier, Kégl (2014) shows that

$$Z(\mathbf{h}, \mathbf{W}) = \frac{e^\alpha + e^{-\alpha}}{2} - \frac{e^\alpha - e^{-\alpha}}{2} \sum_{\ell=1}^{K} v_\ell \big(\mu_{\ell+} - \mu_{\ell-}\big), \tag{6}$$

where

$$\mu_{\ell-} = \sum_{i=1}^{n} w_{i,\ell} \mathbb{I}\{\varphi(\mathbf{x}_i) \neq y_{i,\ell}\} \tag{7}$$

is the *weighted per-class error rate* and

$$\mu_{\ell+} = \sum_{i=1}^{n} w_{i,\ell} \mathbb{I}\{\varphi(\mathbf{x}_i) = y_{i,\ell}\} \tag{8}$$

is the *weighted per-class correct classification rate* for each class $\ell = 1, \ldots, K$. The quantity

$$\gamma_\ell = v_\ell \big(\mu_{\ell+} - \mu_{\ell-}\big) = \sum_{i=1}^{n} w_{i,\ell} v_\ell \varphi(\mathbf{x}_i) y_{i,\ell} \tag{9}$$

is the *classwise edge* of $\mathbf{h}(\mathbf{x})$. The full multi-class edge of the classifier is then

$$\gamma = \gamma(\mathbf{v}, \varphi, \mathbf{W}) = \sum_{\ell=1}^{K} \gamma_\ell = \sum_{\ell=1}^{K} v_\ell \big(\mu_{\ell+} - \mu_{\ell-}\big) = \sum_{i=1}^{n} \varphi(\mathbf{x}_i) \sum_{\ell=1}^{K} w_{i,\ell} v_\ell y_{i,\ell}. \tag{10}$$

With this notation, the classical (Freund and Schapire, 1997) binary coefficient α is recovered: it is easy to see that (6) is minimized when

$$\alpha = \frac{1}{2} \log \frac{1+\gamma}{1-\gamma}. \tag{11}$$

With this optimal coefficient, similarly to the binary case, (6) becomes $Z(\mathbf{h}, \mathbf{W}) = \sqrt{1-\gamma^2}$, so $Z(\mathbf{h}, \mathbf{W})$ is minimized when γ is maximized. From (10) it then follows that $Z(\mathbf{h}, \mathbf{W})$ is minimized if v_ℓ agrees with the sign of $\big(\mu_{\ell+} - \mu_{\ell-}\big)$, that is,

$$v_\ell = \begin{cases} 1 & \text{if } \mu_{\ell+} > \mu_{\ell-} \\ -1 & \text{otherwise} \end{cases} \tag{12}$$

for all classes $\ell = 1, \ldots, K$.

As in the binary case, it can be shown that if $\gamma > \delta$, the training error becomes zero exponentially fast, more precisely, after

$$T^* = \left\lceil \frac{2\log\big(2n(K-1)\big)}{\delta^2} \right\rceil + 1 \tag{13}$$

iterations.[2] The problem, and the subject of this open problem paper, is that $\gamma > \delta$ is not implied by the weak learning condition on our single binary classifier φ. To show this, let us rewrite the edge (10) as

$$\gamma = \sum_{i=1}^{n} \varphi(\mathbf{x}_i)(w_i^+ - w_i^-) = \sum_{i=1}^{n} \varphi(\mathbf{x}_i)\text{sign}(w_i^+ - w_i^-)|w_i^+ - w_i^-|,$$

where

$$w_i^+ = \sum_{\ell=1}^{K} w_{i,\ell}\mathbb{I}\left\{v_\ell y_{i,\ell} = +1\right\},$$

and

$$w_i^- = \sum_{\ell=1}^{K} w_{i,\ell}\mathbb{I}\left\{v_\ell y_{i,\ell} = -1\right\}.$$

By assigning pseudo-labels $y_i' = \text{sign}(w_i^+ - w_i^-)$ and pseudo-weights $w_i' = |w_i^+ - w_i^-|$ to each point $(\mathbf{x}_i, y_i)$, we can rewrite the multi-class edge (10) as a classical binary classification edge $\gamma = \sum_{i=1}^n w_i'\varphi(\mathbf{x}_i)y_i'$. The trouble is that

$$w_\Sigma' \triangleq \sum_{i=1}^{n} w_i' \leq 1. \tag{14}$$

The classical weak learning condition requires that there exists a constant $\delta' > 0$ for which, with any weighting $\mathbf{w}$ summing to one $\sum_{i=1}^n w_i = 1$, there exists a φ with edge

$$\gamma \triangleq \sum_{i=1}^{n} w_i\varphi(\mathbf{x}_i)y_i \geq \delta'. \tag{15}$$

Since $w_\Sigma' \leq 1$, this only implies $\delta = \delta' w_\Sigma' \leq \delta'$ in (13). If w_Σ' can be arbitrarily small, then the convergence to zero training error can be arbitrarily slow. The open problem is to find a lower bound on w_Σ' which may depend on the number of classes K but is *independent of the number of training points* n.

Acknowledgments

This work was supported by the ANR-2010-COSI-002 grant of the French National Research Agency.

[2] This bound can be tightened to $\left\lceil \frac{2\log\left(n\sqrt{K-1}\right)}{\delta^2} \right\rceil + 1$ (Schapire and Singer, 1999).

Appendix A. Remarks

Remark 1 *The reason why the proof works out in the binary case $K = 2$ is that we have strict equality in (14). Indeed, transforming the binary setup into multi-class with two label columns, we always have $y_{i,1} = -y_{i,2}$ and $v_1 = -v_2$, which means that either $w_i^+ = 0$ (when $\mathbf{v} = -\mathbf{y}_i$) or $w_i^- = 0$ (when $\mathbf{v} = \mathbf{y}_i$) so we end up with $w_i' = w_{i,1} + w_{i,2}$ for all $i = 1, \ldots, n$, and $\sum_{i=1}^n w_i' = \sum_{i=1}^n (w_{i,1} + w_{i,2}) = 1$ in each iteration (by classical boosting-type induction argument).*

Remark 2 *The reason why we do not face this problem in Schapire and Singer (1999)'s original* ADABOOST.MH *setup is that $\mathbf{h}(\mathbf{x})$ is not factorized (5) there, rather, it is assumed that K independent one-against-all classifiers $\varphi_\ell(\mathbf{x})$, $\ell = 1, \ldots, K$ are trained separately on the binary problems defined by the columns of $\mathbf{Y}$ and $\mathbf{W}$. The classwise edges*

$$\gamma_\ell = \sum_{i=1}^n w_{i,\ell} \varphi_\ell(\mathbf{x}_i) y_{i,\ell}$$

are not coupled by the label-independent single classifier $\varphi(\mathbf{x})$ as in (9). From the weak learning assumption (15) it follows that for each classwise edge we have $\gamma_\ell \geq \delta' \sum_{i=1}^n w_{i,\ell}$, and so

$$\gamma = \sum_{\ell=1}^K \gamma_\ell \geq \delta' \sum_{i=1}^n \sum_{i=\ell}^n w_{i,\ell} = \delta'.$$

Remark 3 *If it is assumed that $\mathbf{v}$ is fixed, than it seems plausible that we can construct a toy example with $w_\Sigma' = 0$. What makes the problem non-trivial is that the base learner optimizes the edge $\gamma(\mathbf{v}, \varphi, \mathbf{W})$ (10) in φ* and *$\mathbf{v}$ simultaneously. In the case of decision stumps (Appendix C), a global optimum can be found in $\Theta(ndK)$ time. If φ is learned by a generic binary classification algorithm, a greedy but practically very efficient iterative optimization usually leads to an excellent solution (Gao and Koller, 2011). So the question is whether there exists a setup ($\mathbf{X}$, $\mathbf{W}$, $\mathbf{Y}$, and function class) in which* all *of the 2^K different vote vectors $\mathbf{v} = \{\pm 1\}^K$ lead to arbitrarily small (or zero) w_Σ', or we can find a constant (independent of n) lower bound ω such that with* at least one *vote vector $\mathbf{v}$ and classifier φ, $w_\Sigma' \geq \omega$ holds. This would then imply $\delta = \delta' \omega$ in (13).*

Remark 4 *It is easy to see that the edge $\gamma(\mathbf{v}, \varphi, \mathbf{W})$ can never be negative, since if it were, flipping the sign of either $\mathbf{v}$ or φ would make it strictly positive. This means that in practice, the algorithm does not get stuck: given even a* random *classifier φ, unless the edge is exactly zero for* all *vote vectors $\mathbf{v}$, we can find a vote vector for which the edge is strictly positive. The only question we raise here is whether this edge can become arbitrarily small (even if the weak learning condition (15) holds).*

Appendix B. The pseudocode of the ADABOOST.MH algorithm with factorized base classifiers

The following is the pseudocode of the ADABOOST.MH algorithm with factorized base classifiers (5). $\mathbf{X}$ is the $n \times d$ observation matrix, $\mathbf{Y}$ is the $n \times K$ label matrix, $\mathbf{W}$ is the user-defined weight matrix used in the definition of the weighted Hamming error (1) and the weighted exponential margin-based error (3), BASE$(\cdot, \cdot, \cdot)$ is the base learner algorithm, and T is the number of iterations.

$\alpha^{(t)}$ is the base coefficient, $\mathbf{v}^{(t)}$ is the vote vector, $\varphi^{(t)}(\cdot)$ is the scalar base (weak) classifier, $\mathbf{h}^{(t)}(\cdot)$ is the vector-valued base classifier, and $\mathbf{f}^{(T)}(\cdot)$ is the final (strong) discriminant function.

```
AdaBoost.MH$(\mathbf{X}, \mathbf{Y}, \mathbf{W}, \textsc{Base}(\cdot, \cdot, \cdot), T)$
    $\mathbf{W}^{(1)} \leftarrow \frac{1}{n}\mathbf{W}$
    for $t \leftarrow 1$ to $T$
        $\big(\alpha^{(t)}, \mathbf{v}^{(t)}, \varphi^{(t)}(\cdot)\big) \leftarrow \textsc{Base}\big(\mathbf{X}, \mathbf{Y}, \mathbf{W}^{(t)}\big)$
        $\mathbf{h}^{(t)}(\cdot) \leftarrow \alpha^{(t)}\mathbf{v}^{(t)}\varphi^{(t)}(\cdot)$
        for $i \leftarrow 1$ to $n$ for $\ell \leftarrow 1$ to $K$
            $w_{i,\ell}^{(t+1)} \leftarrow w_{i,\ell}^{(t)} \frac{e^{-h_\ell^{(t)}(\mathbf{x}_i) y_{i,\ell}}}{\underbrace{\sum_{i'=1}^{n}\sum_{\ell'=1}^{K} w_{i',\ell'}^{(t)} e^{-h_{\ell'}^{(t)}(\mathbf{x}_{i'}) y_{i',\ell'}}}_{Z\left(\mathbf{h}^{(t)}, \mathbf{W}^{(t)}\right)}}$
    return $\mathbf{f}^{(T)}(\cdot) = \sum_{t=1}^{T} \mathbf{h}^{(t)}(\cdot)$
```

Appendix C. Multi-class decision stumps

The simplest scalar base learner used in practice on numerical features is the *decision stump*, a one-decision two-leaf decision tree of the form

$$\varphi_{j,b}(\mathbf{x}) = \begin{cases} 1 & \text{if } x^{(j)} \geq b, \\ -1 & \text{otherwise,} \end{cases}$$

where j is the index of the selected feature and b is the decision threshold. If the feature values $\left(x_1^{(j)}, \ldots, x_n^{(j)}\right)$ are pre-ordered before the first boosting iteration, a decision stump maximizing the edge (10) (or minimizing the energy (5)[3]) can be found very efficiently in $\Theta(ndK)$ time.

The pseudocode of the algorithm is given in Figure 2. STUMPBASE first calculates the edge vector $\gamma^{(0)}$ of the constant classifier $\mathbf{h}^{(0)}(\mathbf{x}) \equiv \mathbf{1}$ which will serve as the initial edge vector for each featurewise edge-maximizer. Then it loops over the features, calls BESTSTUMP to return the best featurewise stump, and then selects the best of the best by minimizing the energy (5). BESTSTUMP loops over all (sorted) feature values $s_1, \ldots, s_{n-1}$. It considers all thresholds b halfway between two non-identical feature values $s_i \neq s_{i+1}$. The main trick (and, at the same time, the bottleneck of the algorithm) is the update of the classwise edges in lines 4-5: when the threshold moves from $b = \frac{s_{i-1}+s_i}{2}$ to $b = \frac{s_i+s_{i+1}}{2}$, the classwise edge γ_ℓ of $\mathbf{1}\varphi(\mathbf{x})$ (that is, $\mathbf{v}\varphi(\mathbf{x})$ with $\mathbf{v} = \mathbf{1}$) can only change by $\pm w_{i,\ell}$, depending on the sign $y_{i,\ell}$ (Figure 1). The total edge of $\mathbf{v}\varphi(\mathbf{x})$ with optimal votes (12) is then the sum of the absolute values of the classwise edges of $\mathbf{1}\varphi(\mathbf{x})$ (line 7).

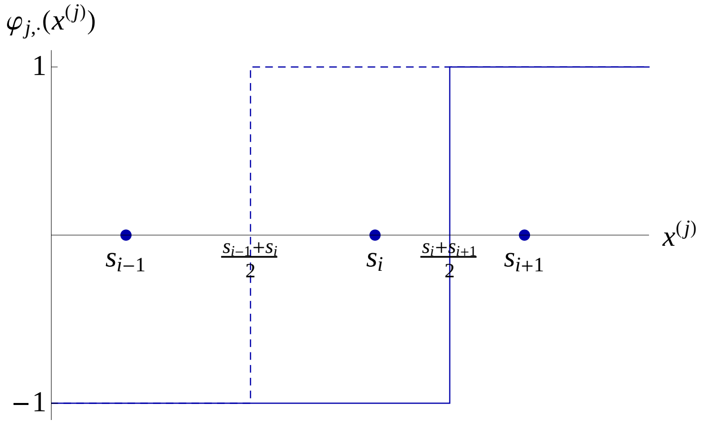

Figure 1: Updating the edge γ_ℓ in line 5 of BESTSTUMP. If $y_{i,\ell} = 1$, then γ_ℓ decreases by $2w_{i,\ell}$, and if $y_i = -1$, then γ_ℓ increases by $2w_{i,\ell}$.

[3] Note the distinction: for full binary $\mathbf{v}$ the two are equivalent, but for ternary or real valued $\mathbf{v}$ and/or real valued $\phi(\mathbf{x})$ they are not. In Figure 2 we are maximizing the edge within each feature (line 7 in BESTSTUMP) but across features we are minimizing the energy (line 7 in STUMPBASE). Updating the energy inside the inner loop (line 4) could not be done in $\Theta(K)$ time.

```
STUMPBASE($\mathbf{X}, \mathbf{Y}, \mathbf{W}$)
1   for $\ell \leftarrow 1$ to $K$    ▷ for all classes
2       $\gamma_\ell^{(0)} \leftarrow \sum_{i=1}^{n} w_{i,\ell} y_{i,\ell}$    ▷ classwise edges (9) of constant classifier $\mathbf{h}^{(0)}(\mathbf{x}) \equiv \mathbf{1}$
3   for $j \leftarrow 1$ to $d$    ▷ all (numerical) features
4       $\mathbf{s} \leftarrow$ SORT$\big(x_1^{(j)}, \ldots, x_n^{(j)}\big)$    ▷ sort the $j$th column of $\mathbf{X}$
5       $(\mathbf{v}_j, b_j, \gamma_j) \leftarrow$ BESTSTUMP$(\mathbf{s}, \mathbf{Y}, \mathbf{W}, \gamma^{(0)})$    ▷ best stump per feature
6       $\alpha_j \leftarrow \frac{1}{2} \log \frac{1+\gamma_j}{1-\gamma_j}$    ▷ base coefficient (11)
7   $j^* \leftarrow \arg\min_j Z\big(\alpha_j \mathbf{v}_j \varphi_{j,b_j}, \mathbf{W}\big)$    ▷ best stump across features
8   return $\big(\alpha_{j^*}, \mathbf{v}_{j^*}, \varphi_{j^*,b_{j^*}}(\cdot)\big)$
```

```
BESTSTUMP($\mathbf{s}, \mathbf{Y}, \mathbf{W}, \gamma^{(0)}$)
1   $\gamma^* \leftarrow \gamma^{(0)}$    ▷ best edge vector
2   $\gamma \leftarrow \gamma^{(0)}$    ▷ initial edge vector
3   for $i \leftarrow 1$ to $n-1$    ▷ for all points in order $s_1 \leq \ldots \leq s_{n-1}$
4       for $\ell \leftarrow 1$ to $K$    ▷ for all classes
5           $\gamma_\ell \leftarrow \gamma_\ell - 2 w_{i,\ell} y_{i,\ell}$    ▷ update classwise edges of stump with $\mathbf{v} = \mathbf{1}$
6       if $s_i \neq s_{i+1}$ then    ▷ no threshold if identical coordinates $s_i = s_{i+1}$
7           if $\sum_{\ell=1}^{K} |\gamma_\ell| > \sum_{\ell=1}^{K} |\gamma_\ell^*|$ then    ▷ found better stump
8               $\gamma^* \leftarrow \gamma$    ▷ update best edge vector
9               $b^* \leftarrow \frac{s_i + s_{i+1}}{2}$    ▷ update best threshold
10  for $\ell \leftarrow 1$ to $K$    ▷ for all classes
11      $v_\ell^* \leftarrow \text{sign}(\gamma_\ell)$    ▷ set vote vector according to (12)
12  if $\gamma^* = \gamma^{(0)}$    ▷ did not beat the constant classifier
13      return $(\mathbf{v}^*, -\infty, \|\gamma^*\|_1)$    ▷ constant classifier with optimal votes
14  else
15      return $(\mathbf{v}^*, b^*, \|\gamma^*\|_1)$    ▷ best stump
```

Figure 2: Exhaustive search for the best decision stump. BESTSTUMP receives a sorted column (feature) $\mathbf{s}$ of the observation matrix $\mathbf{X}$. The sorting in line 4 can be done once for all features outside of the boosting loop. BESTSTUMP examines all thresholds b halfway between two non-identical coordinates $s_i \neq s_{i+1}$ and returns the threshold b^* and vote vector $\mathbf{v}^*$ that maximizes the edge $\gamma(\mathbf{v}, \varphi_{j,b}, \mathbf{W})$. STUMPBASE then sets the coefficient α_j according to (11) and chooses the stump across features that minimizes the energy (4).

References

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*, 55:119–139, 1997.

T. Gao and D. Koller. Multiclass boosting with hinge loss based on output coding. In *International Conference on Machine Learning*, 2011.

B. Kégl. The return of AdaBoost.MH: multi-class Hamming trees. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6086.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.