IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 7, 2020, accepted July 26, 2020, date of publication August 3, 2020, date of current version August 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3013908

KalmanTune: A Kalman Filter Based Tuning Method to Make Boosted Ensembles Robust to Class-Label Noise

ARJUN PAKRASHI AND BRIAN MAC NAMEE
Insight Centre for Data Analytics, School of Computer Science, University College Dublin, Dublin 4, D04 V1W8 Ireland
Corresponding author: Arjun Pakrashi (arjun.pakrashi@ucd.ie)

This work was supported by the Science Foundation Ireland (SFI) under Grant SFI/12/RC/2289_P2.

ABSTRACT Boosting has been shown to be a very effective approach to training ensemble classification models. Although they perform very well, boosting algorithms are sensitive to class-label noise (where training data instances are mislabelled). As the level of class-label noise in the training dataset increases, the generalisation performance of ensembles trained using boosting decreases. This paper introduces KalmanTune, a tuning process that can be applied to ensemble models after they have been trained using a boosting algorithm that reduces the impact of class-label noise. KalmanTune frames the tuning of a trained ensemble model as a static state estimation problem that can be addressed using a Kalman filter. This approach exploits the sensor fusion capability of the Kalman filter to reduce the impact of class-label noise on the trained ensemble. This paper describes KalmanTune and an evaluation experiment performed using 34 multi-class datasets with 5 levels of synthetically induced class-label noise that demonstrates that applying KalmanTune after training can improve the performance of ensemble models trained using boosting, especially when training data contains noisy class-labels.

INDEX TERMS Multi-class, classification, ensemble, Kalman filter.

I. INTRODUCTION

Boosting is an ensemble method that has been shown to perform very well for multi-class classification problems. A boosting algorithm (*AdaBoost* [1] is the most well-known) iteratively trains an ensemble of component classifiers, where at each iteration of the training process, t, the component classifier, h_t, is trained in such a way that it pays particular attention to the datapoints that were misclassified by the classifiers trained in the previous iterations. In this way classifiers in a boosted ensemble are trained to specialise in particular areas of the input feature space. After they have been trained the outputs of the component classifiers for a query instance can be combined to give the overall output of the ensemble. It is known, however, that boosting algorithms are sensitive to *class-label noise* [2], in other words, incorrectly labelled datapoints in the training dataset. As the amount of class-label noise increases, the performance of boosting decreases [3]. Although several modifications and extensions to the original AdaBoost algorithm have been proposed to be more robust to class-label noise, for example *modified AdaBoost* (or *MAdaBoost*) [4], sensitivity to class-label noise remains a problem for boosting approaches in general.

This paper introduces KalmanTune, a tuning procedure that can be applied to ensemble classifiers after they have been trained using a boosting algorithm to make them more robust to noisy class labels. KalmanTune uses a Kalman filter [5], [6] to reweight the contributions of component classifiers in an ensemble so that the influence of component classifiers that have been impacted by noisy class labels is reduced. KalmanTune can be applied as a tuning step to classifier ensembles trained using any boosting algorithm. This paper describes KalmanTune in detail and demonstrates, through a large evaluation experiment, how it can improve the performance of classifier ensembles trained using the AdaBoost and MAdaBoost boosting algorithms.

The paper is structured as follows. Section II describes popular boosting methods and the Kalman filter. Section III introduces the KalmanTune method. The design of the evaluation experiment performed is described in Section IV and the

The associate editor coordinating the review of this manuscript and approving it for publication was Victor S. Sheng.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

results of this experiment are discussed in Section V. Finally, Section VI concludes the paper and suggests directions for future work.

II. RELATED WORK

This section first introduces boosting and variants of AdaBoost designed to be more robust to class-label noise. Focus then turns to an introduction to the nessessary aspects of Kalman filters, which are the basis of KalmanTune, and a description of how Kalman filters can be used for static state estimation.

A. BOOSTING

Boosting is an ensemble learning method which combines multiple weak learners to make a combined strong learner. Boosting trains multiple component classifiers, h_t, in a sequence and then combines their outputs to make classifications. At each training iteration, t, the component classier, h_t, pays more attention to the training datapoints incorrectly classified by models trained at the previous iterations. The specific details of how component classifiers are combined and how the training datapoints weighted at each iteration differentiate the variety of boosting algorithms in the literature.

AdaBoost [1] is the most well-known boosting algorithm. AdaBoost.SAMME is a multi-class extension of AdaBoost [7], and is the specific version of AdaBoost used throughout this paper (AdaBoost and AdaBoost.SAMME will be used synonymously). AdaBoost trains individual learners, h_t, at each iteration t and performs a weighted combination of them to find an ensemble model $H_T(\boldsymbol{x})$, as defined in the following equation:

$$H_T(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t \mathbb{1}(h_t(\boldsymbol{x}) = c) \tag{1}$$

where T is the total number component learners in the ensemble, and the α_t values are computed to minimise an exponential loss function. Each $h_t = \mathcal{L}(\mathcal{D}, \boldsymbol{w}_t)$ is trained using a learner $\mathcal{L}$, using the training dataset $\mathcal{D}$ under a distribution $\boldsymbol{w}_t$, where each $\boldsymbol{w}_t(\boldsymbol{x})$ indicates the weight associated with the training datapoint $\boldsymbol{x} \in \mathcal{D}$ at iteration t. The weights are computed as follows:

$$B_t(\boldsymbol{x}) = \prod_{i=1}^{t} \exp(-\alpha_i(class(h_i(\boldsymbol{x})) \neq class(\boldsymbol{x}))) \tag{2}$$

$$\boldsymbol{w}_{t+1}(\boldsymbol{x}) = \frac{1}{\boldsymbol{Z}_t} \begin{cases} \boldsymbol{w}_1(\boldsymbol{x}) B_t(\boldsymbol{x}) & \text{AdaBoost} \\ \boldsymbol{w}_1(\boldsymbol{x}) \min\{1, B_t(\boldsymbol{x})\} & \text{MAdaBoost} \end{cases} \tag{3}$$

where *class* indicates the class membership of the prediction scores and $\boldsymbol{Z}_t$ is a normalisation term.

In the case of AdaBoost, if a specific datapoint is misclassified, then the subsequent iteration will weight those datapoints more following Eq. (2) and (3). For those datapoints which are very difficult to fit, in the subsequent iterations of AdaBoost the weights of the datapoints would continually increase. If there are outliers, or the training data contains mislabelled datapoints (class-label noise), then in an attempt to classify the noisy datapoints correctly as mentioned before, AdaBoost will continually increase their weights without any bounds. This results in poor generalisation performance. To accomodate the mislabelled datapoints and the outliers, the decision boundary becomes complicated, compomising generalisation performance of the model. The main contributor of this phenomenon is due to the exponential loss function, which AdaBoost optimises. An interesting analysis and study on class-label noise robustness of AdaBoost can be found in [8].

To overcome this noise sensitivity, MAdaBoost [4] was proposed. The main change is simple, MAdaBoost simply places an upper bound on the weights that can be assigned to any datapoint (Eq. (3)). MAdaBoost has been shown to be very robust to noise and to perform very well, when compared to other methods, in the presence of class-label noise [9]. A detailed theoretical analysis of the class-label noise robustness of MAdaBoost is given by Domingo and Watanabe [4].

There are other approaches similar to MAdaBoost that make boosting more robust to class-label noise. For example, FilterBoost [10] optimises the log loss function, leading to a weight update rule which caps the weight upper bound of a datapoint to 1 using a smooth function. BrownBoost [2], developed for binary classification, is an improvement on AdaBoost that optimises a non-convex loss function making it robust to class-label noise. Also, BrownBoost "*gives-up on*" specific hard to learn datapoints thus overcoming the noise sensitivity of AdaBoost. Instead of the exponential loss function used in AdaBoost, LogitBoost [11] minimises the logistic loss function, and is less sensitive to noise than AdaBoost. AdaBoost$_{reg}$ [12] introduces soft-margin classification by regularising AdaBoost. SavageBoost [13] proposes a new loss function which does not let the penalty increase as fast as in AdaBoost.

B. KALMAN FILTERS

The discrete Kalman filter is a mathematical framework to estimate an unobservable state of a linear stochastic discrete time controlled process through noisy measurements [14]. Let there be a state, $\boldsymbol{x}$, of a linear stochastic system to be estimated, where $\boldsymbol{x}$ cannot be observed directly. The state of the system is estimated in two ways. Firstly, given an estimate of the state at time step $(t-1)$, $\hat{\boldsymbol{x}}_{t-1}$, a linear model can be used to make an *a priori* state estimate $\hat{\boldsymbol{x}}_t^-$. This is known as the *time update* step, and the uncertainty associated with this estimate is known as the *process error*. Secondly, an external sensor can be used to get an estimate of the state through a *measurement*, z_t, of the system, which would similarly involve a related uncertainty, the *measurement error*.

The Kalman filter combines these two noisy state estimates, the *a priori* estimate, $\hat{\boldsymbol{x}}_t^-$, and the *measurement*, z_t, optimally to generate an *a posteriori* state estimate, $\hat{\boldsymbol{x}}_t$, which potentially has a lower uncertainty than the previous two. This is known as the *measurement update* step.

The equations for the *time update* steps are as follows[1]

$$\hat{\boldsymbol{x}}_t^- = \boldsymbol{A}_t \hat{\boldsymbol{x}}_{t-1} \tag{4}$$

$$\boldsymbol{P}_t^- = \boldsymbol{A}_t \boldsymbol{P}_{t-1} \boldsymbol{A}_t^T + \boldsymbol{Q}_t \tag{5}$$

were $\hat{\boldsymbol{x}}_t^-$ is the *a priori* state estimate; $\boldsymbol{P}_t^-$ is the related covariance matrix representing the uncertainty; $\hat{\boldsymbol{x}}_{t-1}$ is the *a posteriori* state estimate; $\boldsymbol{P}_t$ is the posterior covariance matrix representing the related uncertainty; $\boldsymbol{A}_t$ is the state transition matrix establishing the linear relationship between time steps; and $\boldsymbol{Q}_t$ denotes the process noise covariance matrix, from which the process noise is assumed to generate from.

The equations for the *measurement update* steps are as follows[1]

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_t^- + \boldsymbol{K}_t(z_t - \hat{\boldsymbol{x}}_t^-) \tag{6}$$

$$\boldsymbol{K}_t = \boldsymbol{P}_t^- (\boldsymbol{P}_t^- + \boldsymbol{R}_t)^{-1} \tag{7}$$

$$\boldsymbol{P}_t = (\boldsymbol{I} - \boldsymbol{K}_t)\boldsymbol{P}_t^- \tag{8}$$

where z_t is the measurement; $\boldsymbol{R}_t$ is the related measurement noise covariance matrix indicating the uncertainty; $\boldsymbol{K}_t$ is called the *Kalman gain*, which optimally combines the *a priori* estimate and the measurement; and I is the identity matrix.

The Kalman filter iterates through the time update and the measurement update steps. At iteration t, the *a priori* estimate is used in the measurement update step to get an *a posteriori* estimate, which is fed back to the time update in the next iteration. Detailed descriptions of the Kalman filter can be found in [5], [6], [14].

C. STATIC STATE ESTIMATION WITH KALMAN FILTERS

Depending on the specific state estimation problem, the time update step, measurement, and other parameters of the Kalman filter are specifically designed. Static state estimation problems do not require a time update step as the state is considered static. For example, consider that the altitude of a steadily flying aircraft has to be estimated via an altimeter. In this case in Eq. (5) $\boldsymbol{A}_t$ is an identity matrix, as the state being estimated—the altitude of the aircraft—is constant (but unknown). This makes the altitude estimate $\hat{\boldsymbol{x}}_{t-1}$ directly propagate to $\hat{\boldsymbol{x}}_t^-$. Also, it is assumed that the measurement from the altimeter and the related error is known. In this case the measurement update step continually combines the noisy measurements. Therefore, at any iteration t the *a posteriori* estimate of the altitude can be thought as a combination of the noisy altimeter outputs, in other words, an *ensemble* of the noisy altimeter outputs.

A very few works in the literature have utilised the sensor fusion properties of the Kalman filter to combine classifier models or apply it on a similar domain, which is not a direct application of time-series problems. In such domains, Kalman filters have been previously used in data clustering [15] and in non-convex metaheuristic optimisation [16], [17]. The idea of training a classifier ensemble using a Kalman filter was first presented by Pakrashi and Mac Namee [18].

[1]Not considering control input and space transformation matrix.

This approach considers the ensemble model being trained as a state to be estimated and each component classifier as a measurement of this state, with the related misclassification error as the measurement noise. The convergence properties of [18] was shown in [19] and an improvement was also proposed. KalmanTune, however, is a much simpler approach that is applied as a tuning step after an ensemble model has been trained using a boosting algorithm. The following section will describe how tuning an ensemble model can be framed as a static state estimation problem that can be solved using a Kalman filter.

III. THE KalmanTune METHOD

Boosting algorithms, such as AdaBoost, train multiple component classifiers, $h_0, h_1, \ldots, h_T$, using datasets sampled from an overall training dataset. The samples are governed by weights in a sampling distribution which is adjusted so that datapoints mislabelled by classifiers trained in the earlier iterations of the process are more likely to be included in training datasets for classifiers trained in the later iterations. When class-label noise exists in a training dataset the mislabelled datapoints are likely to receive large weights in the sampling distribution as they are likely to be misclassified at multiple iterations in the boosting process. This can result in component classifiers fixated on the mislabelled examples in the training set at the expense of low generalisation error. The KalmanTune method exploits the sensor fusion property of the Kalman filter to enhance the generalisation performance of ensemble classifiers trained using boosting, especially when these ensembles are trained using datasets containing noisy class-labels.

Framing a problem as one of static state estimation performed using a Kalman filter requires (1) the definition of the state being estimated, (2) a sequence of measurements that will be used to estimate the state, and (3) an uncertainty associated with each of these measurements. In KalmanTune these components are defined as follows:

1) the state to be estimated is the final ensemble classifier;
2) the sequence of measurements is the sequence of models created during the boosting process, where the measurement at iteration t is $z_t = \sum_{i=1}^{t} \alpha_t h_t(\boldsymbol{x})$;
3) the misclassification error recorded by the ensemble model at iteration t on the training dataset is taken as the uncertainty of the measurement at iteration t.

The application of the Kalman filter results in a set of Kalman gain values which are used to re-weighting the outputs of the component classifiers in an ensemble model. Eq. (6) decides how much of the difference between the present *a priori* estimate and the measurement should be incorporated to get the *a posteriori* estimate, through the Kalman gain. The Kalman gain essentially performs a weighted average, but this weight is decided based on the uncertainties of the time update step and the measurement process as in Eq. (7). If the uncertainty related to the *a priori* stage is less than the measurement uncertainty, then Kalman gain is lower,

therefore weighing the measurement less. If the measurement has a lower uncertainty, then it is weighted more.

In KalmanTune the state to be estimated is the ensemble model itself. The state of a typical Kalman filter is represented as a numerical scalar or vector and so a state representation to capture an ensemble model is required. In KalmanTune a state estimates an ensemble model which is represented as the set of prediction scores given by that model to each datapoint in the training dataset. This is illustrated in Table 1.

TABLE 1. Intermediate representation of a state for KalmanTune. A trained model is represented using the prediction scores of a given set of datapoints. This representation is used in $y_t^{(a)}$ and $y_t^{(k)}$ in Eq. (9) and Eq. (12).

Representation of a state $\hat{\boldsymbol{y}}_t$			
	c_1	c_2	c_3
$\boldsymbol{x}_1$	0.10	0.89	0.01
$\boldsymbol{x}_2$	0.08	0.27	0.65
⋮	⋮	⋮	⋮
$\boldsymbol{x}_n$	0.77	0.20	0.03

Following the outline of the Kalman filter given in Section II-B the update steps for KalmanTune are:[2]

$$\hat{\boldsymbol{y}}_t^{(a)} = \sum_{j=1}^{t} \alpha_j h_j(\boldsymbol{x}) \tag{9}$$

$$r_t = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \mathbb{1}(class(\boldsymbol{x}) \neq class(\hat{\boldsymbol{y}}_t^{(a)})) \tag{10}$$

$$k_t = \frac{p_t}{(p_t + r_t)} \tag{11}$$

$$\hat{\boldsymbol{y}}_{t+1}^{(k)} = \hat{\boldsymbol{y}}_t^{(k)} + k_t(\hat{\boldsymbol{y}}_t^{(a)} - \hat{\boldsymbol{y}}_t^{(k)}) \tag{12}$$

$$p_{t+1} = (1 - k_t)p_t \tag{13}$$

$$p_{t+1} = \max\{0, \min\{1, p_{t+1} + \mu\}\} \tag{14}$$

Eq. (12) corresponds to Eq. (6), where $\hat{\boldsymbol{y}}_t^{(k)}$ is the intermediate representation of the estimated state by the Kalman filter, and $\hat{\boldsymbol{y}}_t^{(a)}$ is the intermediate representation of the tth boosting ensemble step output, the measurement, as shown in Eq. (9). Eq. (11) corresponds to Eq (7). r_t in Eq. (10) is set as the misclassification rate of H_t. A fixed amount of process noise μ is included in Eq. (14) to stop the Kalman filter from converging too fast. In Eq. (14) μ is a very small real value sampled from a uniform distribution. The combination of *min* and *max* operation is included such that after adding the value μ, the expression $p_{t+1} + \mu$ does not drop below 0 or exceed 1.

Note that, as the KalmanTune is applied to an already learned boosting classifier model, threfore its performance will depend on how those models perform. The target of KalmanTune is to stop sudden changes in adjustments in boosting methods. Therefore as the iterations proceed, the effects of the measurements becomes much lesser if the underlying booster starts converging. Therefore as more components are added, in this case KalmanTune does not change much. But in the noisy class-label cases, if at a later iteration the underlying booster algorithm makes a sudden change because of class label noise, then KalmanTune will stop this big (and unwanted) changed to be incorporated into the final model.

Algorithm 1 formally defines the KalmanTune process. The ensemble model at step t (Eq. (9)) is used as the measurement and no feedback goes to the boosting algorithm. Therefore, the proposed method can be used as a post-training tuning step applied after an ensemble model has been trained using a boosting algorithm. An illustration of KalmanTune training is provided in Figure 1, which shows the interaction of the Kalman filter and the boosting algorithm.

Algorithm 1 KalmanTune Training

1: **procedure** train($\mathcal{D}$, $\{h_t, \alpha_t | \forall_{1 \leq t \leq T}\}$, μ, T)
2: $p_1 = 1, t = 1$
3: **for** $t \leq T$ **do**
4: $\hat{\boldsymbol{y}}_t^{(a)} = \sum_{j=1}^{t} \alpha_j h_j(\boldsymbol{x}) \forall_{\boldsymbol{x} \in \mathcal{D}}$ ▷ Measurement
5: $r_t = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \mathbb{1}(class(\boldsymbol{x}) \neq class(\hat{\boldsymbol{y}}_t^{(a)}))$
6: $k_t = \frac{p_t}{(p_t + r_t)}$ ▷ Kalman gain
7: $p_{t+1} = (1 - k_t)p_t$
8: $p_{t+1} = \max\{0, \min\{1, p_{t+1} + \mu\}\}$
9: **end for**
10: **return** $(\{h_t, \alpha_t, k_t | \forall_{1 \leq t \leq T}\})$
11: **end procedure**

The objective of the training process is to compute the Kalman gains which can used (following Eq. (12)) to make predictions using the ensemble applied to query datapoints. Algorithm 2 shows the prediction algorithm, which is much simpler than the training algorithm. KalmanTune makes predictions based on the trained boosting model as well as the Kalman gains from the training step. At every iteration it first gets the measurement in line 4, and then it uses k_t, the Kalman gain computed during training, to combine the measurement. Finally, it returns the class prediction of the given dataset. In this case in line 5 is the KalmanTune estimate. The KalmanTune estimate $\hat{\boldsymbol{y}}_T^{(k)}$ is expected to be more noise robust than $\hat{\boldsymbol{y}}_T^{(a)}$.

As previously mentioned, KalmanTune can be used with any boosting algorithm. KalmanTune should improve the on the performance of the bare boosting algorithm, especially when the base algorithm is sensitive to class-label noise. The next section describes an experiment that evaluates the impact of post-training tuning using KalmanTune on ensemble models trained using the AdaBoost and MAdaBoost boosting algorithms.

IV. EXPERIMENT

To asses the effectiveness of KalmanTune at improving the performance of boosted ensemble models, an evaluation

[2]The lower case symbols in the equations indicate scalars which correspond to the uppercase symbols used in the more generic outline given in Section II-B.

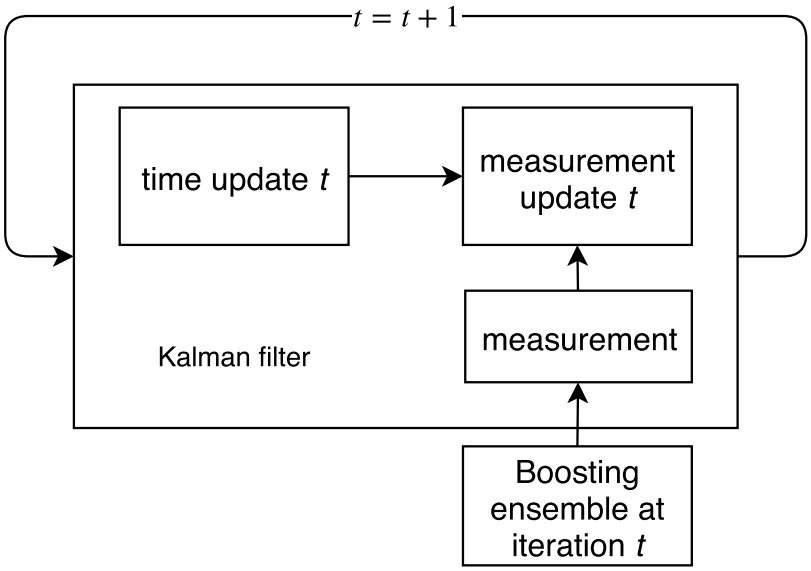

FIGURE 1. An illustration of KalmanTune, showing the interaction of the Kalman filter and boosting.

Algorithm 2 KalmanTune Prediction

procedure predict($\boldsymbol{x}$, $\{h_t, \alpha_t, k_t | \forall_{1 \leq t \leq T}\}$, T)
  $\hat{\boldsymbol{y}}_1^{(k)} = [0], t = 1$
  for $t \leq T$ **do**
    $\hat{\boldsymbol{y}}_t^{(a)} = \sum_{j=1}^{t} \alpha_j h_j(\boldsymbol{x})$ ▷ Measurement
    $\hat{\boldsymbol{y}}_{t+1}^{(k)} = \hat{\boldsymbol{y}}_t^{(k)} + k_t(\hat{\boldsymbol{y}}_t^{(a)} - \hat{\boldsymbol{y}}_t^{(k)})$ ▷ KF Update
  end for
  return $(\hat{\boldsymbol{y}}_T^{(k)})$
end procedure

experiment was performed using 34 well-known multi-class datasets from the UCI machine learning repository [20]. The main objective of the experiment to understand if the KalmanTune stage can improve the noise robustness of the given trained ensemble models. The used datasets are listed in Table 2. For each dataset, 4 new modified versions of the original (0% noise) were induced synthetically containing different levels of class-label noise: 5%, 10%, 15% and 20%. To generate noisy class-labels, a fraction of the datapoints were randomly selected and their labels were switched to a randomly selected classes (both from a uniform distribution). This helps control the fraction of class-labels which are incorrectly assigned. The reason for simulating the random noise in the class assignment was to simulate real life mislabelling of dataset. For example, during data collection and manual labelling many datapoints can be mislabelled. Note that the noise is induced only in the class assignments and not the input space.

In the experiment ensemble models were first trained using AdaBoost [1] and MAdaBoost [4]. The KalmanTune tuning process was then applied to each ensemble model and a new set of predictions using the tuned ensemble were generated and evaluated. Note that as KalmanTune tuning is applied after training a model there was no need to retrain the ensembles before applying KalmanTune tuning. AdaBoost was selected because it is sensitive to class-label noise and MAdaBoost was selected because it was designed specifically to be robust to class-label noise. This selection would

TABLE 2. The datasets used in this paper.

dataset names	#datapoints	#dimensions	#classes
mushroom	8124	22	2
iris	150	5	3
glass	214	10	6
car_eval	1728	7	4
cmc	1473	10	3
tvowel	871	4	6
balance_scale	625	5	3
breasttissue	106	10	6
german	1000	21	2
ilpd	579	11	2
ionosphere	351	34	2
knowledge	403	6	4
vertebral	310	7	2
sonar	208	61	2
diabetes	145	4	3
skulls	150	5	5
physio	464	37	3
flags	194	30	8
bupa	345	7	2
cleveland	303	14	5
haberman	306	4	2
hayes-roth	132	6	3
monks	432	7	2
newthyroid	432	7	3
yeast	1484	9	10
spam	4601	58	2
lymphography	148	19	4
movement_libras	360	91	15
SAheart	462	10	2
zoo	101	17	7

allow evaluation of the ability of KalmanTune to improve both a noise-sensitive and noise-robust boosting method. In the results that follow KalmanTune-A refers to an ensemble trained using AdaBoost and tuned using KalmanTune, and KalmanTune-M refers to a model trained using MAdaBoost and tuned using KalmanTune. For all datasets ensemble models are also trained using Bagging [21] so that the performance of these models, which are known to be robust to class-label noise [3], can be used as a baseline against which to compare the performance of the base boosted ensembles and their tuned versions.

For all experiments a 20 times 4-folds cross validation was performed using each algorithm, on each datatset, for each level of class-label noise. The performance measure used throughout is macro-averaged F-Score [22] as some datasets have very heavily imbalanced class label distributions. For KalmanTune the hyperparameter $\mu = u/10^{-4}$, where u is sampled from standard normal distribution. All the algorithms used 100 ensemble components where each component classifier was a CART [22] model.

V. RESULTS

The objective of the experiment is to demonstrate the effectiveness of the KalmanTune step after it is applied on boosting models. In Secton V-A, AdaBoost, which is sensitive to class label noise, will be compared with the performance of KalmanTune-A which improves the model learned by AdaBoost. Next in Section V-B, MAdaBoost, which is robust to class label noise, will be compared with KalmanTune-M which improves the model learned by MAdaBoost. Finally, an overall view will be given in Section V-C.

TABLE 3. Each row indicates the average rank of KalmanTune-A and AdaBoost, the *p*-values of two-tailed Wilcoxon's signed rank sum tests, and the win/lose/tie ratio of KalmanTune-A against AdaBoost for a specific noise level. Different significance thresholds are indicated as *:$\alpha = 0.1$, **:$\alpha = 0.05$, *:$\alpha = 0.01$.**

Noise Level	KalmanTune-A avg. rank	AdaBoost avg. rank	*p*-value	win/lose/tie
0%	1.46	1.54	0.1692	10/7/17
5%	1.44	1.56	0.0857 *	16/12/6
10%	1.28	1.72	0.0009 ***	22/7/5
15%	1.28	1.72	0.0004 ***	22/7/5
20%	1.29	1.71	0.0001 ***	22/8/4

A. AdaBoost & KalmanTune-A

To understand performance of the KalmanTune step when applied on a trained AdaBoost model, KalmanTune-A and AdaBoost model performances are compared. A summary of the results of all the 5 levels of class-label noise over 34 datasets is shown in Table 3. This table is derived from Table 5 to 9 in the Appendix. The higher average rank achieved by KalmanTune-A and the win/lose/tie count shows that KalmanTune tuning has improved the generalisation error of the base AdaBoost ensemble.

As the noise level increases, KalmanTune-A achieves higher average ranks. Note that the number of ties at 0% noise level is very high. This is because when no class-label noise is present in the dataset KalmanTune tuning perform minimal or no modifications to the existing base ensemble model. As the noise level increases, the number of ties decreases drastically and the number of wins for KalmanTune-A increases as the aggregation of the base ensemble model is reweighted to take account of perceived class-label noise.

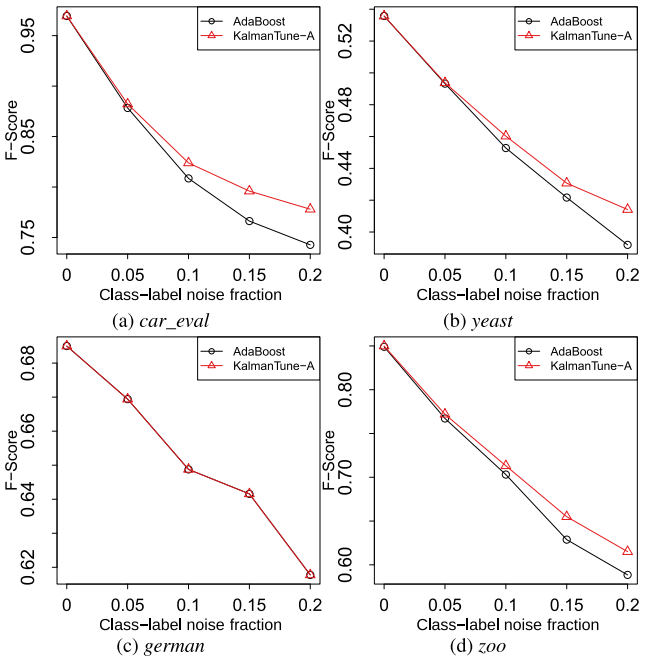

FIGURE 2. F-Score change (crossvalidated) with respect to class-label noise for *car_eval*, *yeast*, *german* and *zoo* datasets. The y-axis is scaled to highlight the differences of the compared algorithms KalmanTune-A and AdaBoost.

To illustrate the impact of adding class-label noise to the test datasets, and to demonstrate the ability of KalmanTune tuning to alleviate this, Figure 2 shows the change in macro-averaged F-Score for the AdaBoost and KalmanTune-A models as the level of induced class-label noise increases for the *car_eval*, *yeast*, *german* and *zoo* datasets (similar patterns are observed in the 30 other datasets). It is clear that, although class-label noise always impacts on model performance, KalmanTune tuning reduces this impact in most cases and rarely leads to a more poorly performing model.

A two tailed Wilcoxon's signed rank sum test following the recommendations of [23] was performed to further evaluate the difference in performance between KalmanTune-A and AdaBoost. The *p*-values arising from this test are shown in Table 3. This shows that with 10%, 15% and 20% class-label noise, the performance of the KalmanTune-A model was significantly better than the base AdaBoost ensemble model, with significance level of $\alpha = 0.01$, and also at 5% class noise level with $\alpha = 0.1$. Although the null hypothesis could not be rejected in the 0% class noise case, KalmanTune-A was able to achieve a better average rank than AdaBoost.

TABLE 4. Each row indicates the average rank of KalmanTune-M and MAdaBoost, the *p*-values of two-tailed Wilcoxon's signed rank sum tests, and the win/lose/tie ratio of KalmanTune-M against MAdaBoost for a specific noise level. Different significance thresholds are indicated as *:$\alpha = 0.1$, **:$\alpha = 0.05$, ***:$\alpha = 0.01$.

Noise Level	KalmanTune-M avg. rank	MAdaBoost avg. rank	*p*-value	win/lose/tie
0%	1.21	1.79	0.0001 ***	26/6/2
5%	1.27	1.74	0.0005 ***	25/9/0
10%	1.38	1.62	0.0569 *	21/13/0
15%	1.35	1.65	0.0056 **	22/12/0
20%	1.35	1.65	0.1757	22/12/0

B. MAdaBoost & KalmanTune-M

A similar comparison was performed between the performance of the KalmanTune-M and MAdaBoost models to understand the improvements made by the KalmanTune step when applied on a trained MAdaBoost model. The summary of the results on all the 5 levels of class-label noise over the 34 datasets is shown in Table 4. This table is derived from Table 10 to 14 in Appendix. The change in average ranks for KalmanTune-M and the win/lose/tie count shows that at lower noise level, overall, KalmanTune-M has performed better than MAdaBoost. Figure 3 shows the change in macro-averaged F-Score with increasing induced class-label noise for KalmanTune-M and MAdaBoost for the *car_eval*, *yeast*, *german* and *zoo* datasets (again, similar patterns are observed in the 30 other datasets).

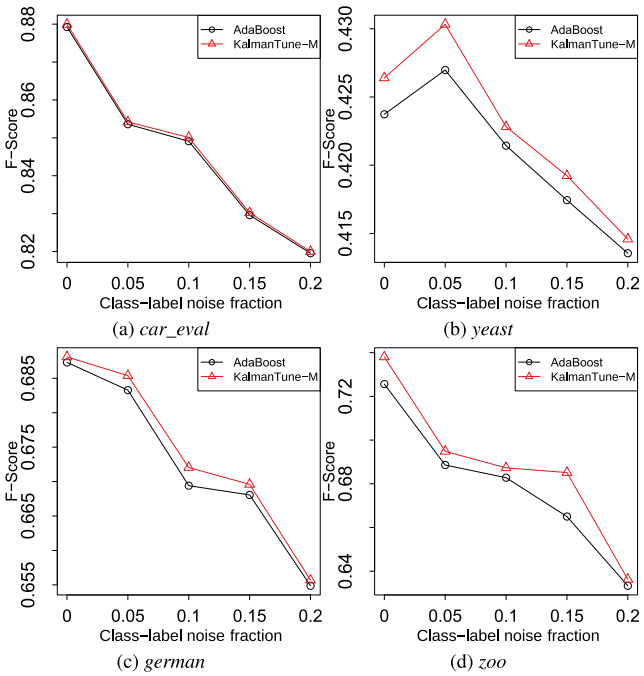

FIGURE 3. F-Score change (crossvalidated) with respect to class-label noise for *car_eval*, *yeast*, *german* and *zoo* datasets. The y-axis is scaled to highlight the differences of the compared algorithms KalmanTune-M and MAdaBoost.

Although KalmanTune-M has always attains a better overall rank, the two tailed Wilcoxon's signed rank sum test

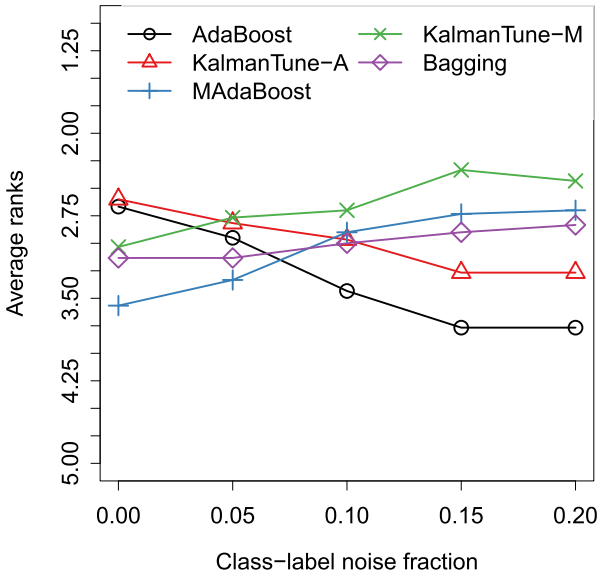

FIGURE 4. Overall average rank change across different noise level. The y-axis is inverted to emphasise that lower values are better.

TABLE 5. AdaBoost and KalmanTune-A F-Scores with 0% label noise.

	AdaBoost	KalmanTune-A
mushroom	1.0000 ± 0.00	1.0000 ± 0.00
iris	0.9434 ± 0.03	0.9434 ± 0.03
glass	0.6968 ± 0.07	0.6965 ± 0.07
car_eval	0.9695 ± 0.02	0.9695 ± 0.02
cmc	0.5067 ± 0.03	0.5133 ± 0.03
tvowel	0.8008 ± 0.04	0.8103 ± 0.03
dermatology	0.9645 ± 0.02	0.9645 ± 0.02
balance_scale	0.6422 ± 0.04	0.6433 ± 0.04
breastcancer	0.9590 ± 0.01	0.9590 ± 0.01
flags	0.3222 ± 0.06	0.3177 ± 0.06
german	0.6850 ± 0.03	0.6850 ± 0.03
wine	0.9709 ± 0.03	0.9709 ± 0.03
ilpd	0.6101 ± 0.04	0.6101 ± 0.04
ionosphere	0.9266 ± 0.02	0.9266 ± 0.02
knowledge	0.9500 ± 0.02	0.9500 ± 0.02
vertebral	0.8001 ± 0.05	0.8001 ± 0.05
sonar	0.8415 ± 0.05	0.8415 ± 0.05
skulls	0.2461 ± 0.07	0.2472 ± 0.07
diabetes	0.9647 ± 0.03	0.9647 ± 0.03
physio	0.9102 ± 0.02	0.9102 ± 0.02
breasttissue	0.6662 ± 0.09	0.6639 ± 0.08
ecoli	0.5882 ± 0.06	0.5888 ± 0.06
bupa	0.6934 ± 0.04	0.6934 ± 0.04
cleveland	0.2963 ± 0.05	0.2975 ± 0.05
haberman	0.5484 ± 0.05	0.5489 ± 0.05
hayes.roth	0.8552 ± 0.05	0.8545 ± 0.05
monks	1.0000 ± 0.00	1.0000 ± 0.00
newthyroid	0.4490 ± 0.04	0.4498 ± 0.04
yeast	0.5358 ± 0.05	0.5357 ± 0.05
spam	0.9501 ± 0.01	0.9501 ± 0.01
lymphography	0.7012 ± 0.18	0.7010 ± 0.18
movement_libras	0.7597 ± 0.04	0.7607 ± 0.04
SAheart	0.6091 ± 0.04	0.6091 ± 0.04
zoo	0.8490 ± 0.11	0.8499 ± 0.11

indicate that the significant differences have reduced as the noise level increases. In this case, KalmanTune-M was significantly better than MAdaBoost at 0%, 5% and 15% noise levels with $\alpha = 0.01$ and 10% noise level with $\alpha = 0.1$.

TABLE 6. AdaBoost and KalmanTune-A F-Scores with 5% label noise.

	AdaBoost	KalmanTune-A
mushroom	0.9964 ± 0.00	0.9963 ± 0.00
iris	0.9173 ± 0.04	0.9187 ± 0.04
glass	0.6606 ± 0.09	0.6594 ± 0.08
car_eval	0.8783 ± 0.04	0.8822 ± 0.04
cmc	0.5051 ± 0.03	0.5109 ± 0.03
tvowel	0.7447 ± 0.04	0.7636 ± 0.04
dermatology	0.9467 ± 0.03	0.9471 ± 0.03
balance_scale	0.6074 ± 0.03	0.6069 ± 0.04
breastcancer	0.9499 ± 0.02	0.9498 ± 0.02
flags	0.3058 ± 0.06	0.3092 ± 0.06
german	0.6694 ± 0.03	0.6694 ± 0.03
wine	0.9528 ± 0.03	0.9528 ± 0.03
ilpd	0.6099 ± 0.04	0.6097 ± 0.04
ionosphere	0.9060 ± 0.03	0.9058 ± 0.03
knowledge	0.9320 ± 0.02	0.9320 ± 0.02
vertebral	0.7770 ± 0.05	0.7770 ± 0.05
sonar	0.8199 ± 0.05	0.8199 ± 0.05
skulls	0.2531 ± 0.07	0.2540 ± 0.07
diabetes	0.9387 ± 0.05	0.9399 ± 0.05
physio	0.8800 ± 0.03	0.8821 ± 0.03
breasttissue	0.6310 ± 0.09	0.6300 ± 0.09
ecoli	0.5654 ± 0.07	0.5632 ± 0.07
bupa	0.6754 ± 0.05	0.6744 ± 0.05
cleveland	0.3006 ± 0.05	0.3047 ± 0.05
haberman	0.5415 ± 0.06	0.5463 ± 0.06
hayes.roth	0.7934 ± 0.08	0.7934 ± 0.08
monks	0.9578 ± 0.02	0.9580 ± 0.02
newthyroid	0.4456 ± 0.04	0.4461 ± 0.04
yeast	0.4933 ± 0.06	0.4938 ± 0.06
spam	0.9259 ± 0.01	0.9243 ± 0.01
lymphography	0.5961 ± 0.14	0.5948 ± 0.14
movement_libras	0.7247 ± 0.04	0.7235 ± 0.04
SAheart	0.5950 ± 0.04	0.5950 ± 0.04
zoo	0.7671 ± 0.12	0.7720 ± 0.12

TABLE 7. AdaBoost and KalmanTune-A F-Scores with 10% label noise.

	AdaBoost	KalmanTune-A
mushroom	0.9957 ± 0.00	0.9952 ± 0.00
iris	0.8776 ± 0.06	0.8774 ± 0.06
glass	0.6294 ± 0.09	0.6250 ± 0.09
car_eval	0.8085 ± 0.05	0.8239 ± 0.05
cmc	0.5010 ± 0.03	0.5065 ± 0.03
tvowel	0.6987 ± 0.05	0.7297 ± 0.04
dermatology	0.9312 ± 0.03	0.9322 ± 0.03
balance_scale	0.5837 ± 0.03	0.5885 ± 0.03
breastcancer	0.9354 ± 0.02	0.9360 ± 0.02
flags	0.2992 ± 0.06	0.2999 ± 0.06
german	0.6488 ± 0.03	0.6488 ± 0.03
wine	0.9299 ± 0.04	0.9299 ± 0.04
ilpd	0.5919 ± 0.05	0.5930 ± 0.05
ionosphere	0.8817 ± 0.04	0.8815 ± 0.04
knowledge	0.9073 ± 0.03	0.9074 ± 0.03
vertebral	0.7607 ± 0.05	0.7607 ± 0.05
sonar	0.7839 ± 0.05	0.7839 ± 0.05
skulls	0.2338 ± 0.05	0.2332 ± 0.06
diabetes	0.8970 ± 0.05	0.8978 ± 0.05
physio	0.8557 ± 0.04	0.8579 ± 0.04
breasttissue	0.6163 ± 0.09	0.6216 ± 0.09
ecoli	0.5476 ± 0.07	0.5484 ± 0.07
bupa	0.6527 ± 0.05	0.6528 ± 0.05
cleveland	0.2987 ± 0.05	0.3006 ± 0.05
haberman	0.5451 ± 0.06	0.5451 ± 0.06
hayes.roth	0.7464 ± 0.09	0.7488 ± 0.09
monks	0.9107 ± 0.03	0.9142 ± 0.04
newthyroid	0.4380 ± 0.04	0.4451 ± 0.05
yeast	0.4528 ± 0.05	0.4602 ± 0.05
spam	0.9103 ± 0.01	0.9081 ± 0.01
lymphography	0.5203 ± 0.13	0.5275 ± 0.13
movement_libras	0.7027 ± 0.05	0.7015 ± 0.04
SAheart	0.5928 ± 0.05	0.5928 ± 0.05
zoo	0.7031 ± 0.11	0.7131 ± 0.12

MAdaBoost is a noise robust improvement of AdaBoost, and was previously shown to be reealtively robust to class-label noise [9]. Despite MAdaBoost's noise resistance, KalmanTune-M was still able to improve the overall results. Although in this case, unlike the previous case, the differences between MAdaBoost and KalmanTune-M slowly reduces as the noise level increases.

C. OVERALL PERFORMANCE

The performance of the different models (trained using AdaBoost, KalmanTune-A, MAdaBoost, KalmanTune-B, and Bagging) on each of the 34 datasets used and the 4 noisy versions of each dataset was measured using macro-averaged F-score and ranked. Average ranks were calculated for each different level of induced class-label noise. Figure 4 shows how the average ranks changed as the level of class-label noise increased. In this comparison it can be seen that MAdaBoost is more robust to class-label noise than AdaBoost. Also, it is clear that in each case using KalmanTune improves performance: KalmanTune-A out-performs AdaBoost and KalmanTune-M outperforms MAdaBoost. Overall KalmanTune-M was able to maintain the best average ranks except at the 0% class-label noise level. Bagging, as expected, started at the lowest rank, emphasising the usefulness of boosting methods, but improved in rank as the level of class-label noise increased (it had equal top rank with KalmanTune-M at the 20% noise level).

These results show that overall, the KalmanTune tuning approach improves the performance of the base boosting models and makes them more robust to class-label noise.

VI. CONCLUSION

This work presented a new approach, KalmanTune, which tunes already trained boosted ensemble models to make them more robust to class-label noise present in the data used to train them. The KalmanTune approach is motivated by the fact that, although they can perform very well, boosting algorithms are sensitive to class-label noise. KalmanTune considers the final ensemble model as a static state to be estimated; the boosted ensemble model at iteration t of the boosting process as a measurement of this state; and the classification error achieved by the model at iteration t as the uncertainty of this measurement. KalmanTune uses a Kalman filter to combine multiple such measurements. The output of this process is a set of Kalman gain values which can be used to aggregate the results of the base models in the ensemble in place of the weighting factors calculated during the boosting process. The application of KalmanTune reduces the impact of abrupt changes incorporated into the ensemble during training. This abrupt changes may occur due

TABLE 8. AdaBoost and KalmanTune-A F-Scores with 15% label noise.

	AdaBoost	KalmanTune-A
mushroom	0.9952 ± 0.00	0.9946 ± 0.00
iris	0.8471 ± 0.07	0.8501 ± 0.07
glass	0.5855 ± 0.09	0.5880 ± 0.09
car_eval	0.7663 ± 0.05	0.7960 ± 0.05
cmc	0.4867 ± 0.03	0.4907 ± 0.03
tvowel	0.6539 ± 0.05	0.7013 ± 0.05
dermatology	0.9143 ± 0.03	0.9150 ± 0.03
balance_scale	0.5727 ± 0.03	0.5813 ± 0.03
breastcancer	0.9209 ± 0.02	0.9221 ± 0.02
flags	0.2994 ± 0.06	0.3004 ± 0.06
german	0.6416 ± 0.03	0.6416 ± 0.03
wine	0.9072 ± 0.04	0.9072 ± 0.04
ilpd	0.5851 ± 0.04	0.5838 ± 0.04
ionosphere	0.8448 ± 0.05	0.8447 ± 0.05
knowledge	0.8826 ± 0.04	0.8841 ± 0.04
vertebral	0.7298 ± 0.06	0.7298 ± 0.06
sonar	0.7404 ± 0.06	0.7404 ± 0.06
skulls	0.2374 ± 0.07	0.2366 ± 0.08
diabetes	0.8440 ± 0.07	0.8459 ± 0.06
physio	0.8255 ± 0.04	0.8296 ± 0.04
breasttissue	0.5678 ± 0.10	0.5746 ± 0.09
ecoli	0.5124 ± 0.06	0.5118 ± 0.06
bupa	0.6382 ± 0.05	0.6386 ± 0.05
cleveland	0.3043 ± 0.05	0.3002 ± 0.05
haberman	0.5352 ± 0.06	0.5388 ± 0.06
hayes.roth	0.6592 ± 0.09	0.6608 ± 0.09
monks	0.8731 ± 0.04	0.8800 ± 0.04
newthyroid	0.4169 ± 0.04	0.4247 ± 0.04
yeast	0.4217 ± 0.04	0.4307 ± 0.04
spam	0.9009 ± 0.01	0.8992 ± 0.01
lymphography	0.4742 ± 0.12	0.4753 ± 0.12
movement_libras	0.6841 ± 0.04	0.6855 ± 0.04
SAheart	0.5846 ± 0.04	0.5846 ± 0.04
zoo	0.6288 ± 0.14	0.6550 ± 0.13

TABLE 9. AdaBoost and KalmanTune-A F-Scores with 20% label noise.

	AdaBoost	KalmanTune-A
mushroom	0.9934 ± 0.00	0.9930 ± 0.00
iris	0.8170 ± 0.07	0.8271 ± 0.07
glass	0.5552 ± 0.09	0.5584 ± 0.09
car_eval	0.7427 ± 0.05	0.7780 ± 0.05
cmc	0.4894 ± 0.03	0.4973 ± 0.03
tvowel	0.6270 ± 0.05	0.6903 ± 0.04
dermatology	0.8846 ± 0.04	0.8857 ± 0.04
balance_scale	0.5621 ± 0.04	0.5717 ± 0.04
breastcancer	0.9060 ± 0.03	0.9097 ± 0.03
flags	0.2786 ± 0.06	0.2792 ± 0.06
german	0.6178 ± 0.03	0.6178 ± 0.03
wine	0.8806 ± 0.05	0.8806 ± 0.05
ilpd	0.5841 ± 0.05	0.5833 ± 0.05
ionosphere	0.7967 ± 0.06	0.7964 ± 0.06
knowledge	0.8363 ± 0.05	0.8383 ± 0.05
vertebral	0.6961 ± 0.06	0.6961 ± 0.06
sonar	0.7181 ± 0.06	0.7181 ± 0.06
skulls	0.2260 ± 0.07	0.2264 ± 0.06
diabetes	0.7955 ± 0.08	0.7965 ± 0.08
physio	0.8044 ± 0.05	0.8100 ± 0.05
breasttissue	0.5287 ± 0.10	0.5326 ± 0.10
ecoli	0.5048 ± 0.07	0.5040 ± 0.07
bupa	0.6359 ± 0.05	0.6366 ± 0.05
cleveland	0.2932 ± 0.06	0.2931 ± 0.06
haberman	0.5354 ± 0.06	0.5380 ± 0.06
hayes.roth	0.6223 ± 0.09	0.6316 ± 0.09
monks	0.8230 ± 0.05	0.8341 ± 0.05
newthyroid	0.4158 ± 0.04	0.4240 ± 0.05
yeast	0.3919 ± 0.05	0.4141 ± 0.05
spam	0.8957 ± 0.01	0.8950 ± 0.01
lymphography	0.4620 ± 0.11	0.4615 ± 0.10
movement_libras	0.6485 ± 0.05	0.6492 ± 0.05
SAheart	0.5791 ± 0.05	0.5791 ± 0.05
zoo	0.5885 ± 0.14	0.6151 ± 0.13

to boosting algorithms attempting to fit a noisy class-label dataset.

TABLE 10. MAdaBoost and KalmanTune-M F-Scores with 0% label noise.

	MAdaBoost	KalmanTune-M
mushroom	1.0000 ± 0.00	1.0000 ± 0.00
iris	0.9460 ± 0.03	0.9467 ± 0.03
glass	0.6286 ± 0.09	0.6382 ± 0.09
car_eval	0.8792 ± 0.03	0.8799 ± 0.03
cmc	0.5303 ± 0.03	0.5297 ± 0.03
tvowel	0.8337 ± 0.03	0.8340 ± 0.03
dermatology	0.9460 ± 0.03	0.9463 ± 0.03
balance_scale	0.5867 ± 0.02	0.5875 ± 0.02
breastcancer	0.9548 ± 0.01	0.9551 ± 0.01
flags	0.2501 ± 0.02	0.2503 ± 0.02
german	0.6873 ± 0.03	0.6881 ± 0.03
wine	0.9454 ± 0.04	0.9456 ± 0.04
ilpd	0.5675 ± 0.04	0.5758 ± 0.04
ionosphere	0.9108 ± 0.03	0.9120 ± 0.03
knowledge	0.9273 ± 0.03	0.9275 ± 0.03
vertebral	0.8040 ± 0.04	0.8035 ± 0.04
sonar	0.7773 ± 0.06	0.7771 ± 0.06
skulls	0.2335 ± 0.07	0.2341 ± 0.07
diabetes	0.9695 ± 0.02	0.9698 ± 0.02
physio	0.9039 ± 0.03	0.9047 ± 0.03
breasttissue	0.6667 ± 0.07	0.6700 ± 0.07
ecoli	0.5107 ± 0.05	0.5176 ± 0.05
bupa	0.6938 ± 0.04	0.6998 ± 0.04
cleveland	0.2730 ± 0.04	0.2749 ± 0.04
haberman	0.5739 ± 0.04	0.5739 ± 0.05
hayes.roth	0.7787 ± 0.09	0.7841 ± 0.09
monks	1.0000 ± 0.00	1.0000 ± 0.00
newthyroid	0.4302 ± 0.03	0.4251 ± 0.03
yeast	0.4237 ± 0.03	0.4264 ± 0.03
spam	0.9332 ± 0.01	0.9359 ± 0.01
lymphography	0.3958 ± 0.03	0.3977 ± 0.03
movement_libras	0.6933 ± 0.05	0.6941 ± 0.05
SAheart	0.6459 ± 0.04	0.6455 ± 0.04
zoo	0.7256 ± 0.09	0.7381 ± 0.10

TABLE 11. MAdaBoost and KalmanTune-M F-Scores with 5% label noise.

	MAdaBoost	KalmanTune-M
mushroom	0.9965 ± 0.00	0.9963 ± 0.00
iris	0.9448 ± 0.03	0.9445 ± 0.03
glass	0.6185 ± 0.09	0.6295 ± 0.08
car_eval	0.8536 ± 0.03	0.8542 ± 0.03
cmc	0.5278 ± 0.03	0.5282 ± 0.02
tvowel	0.8219 ± 0.02	0.8245 ± 0.02
dermatology	0.9416 ± 0.03	0.9415 ± 0.03
balance_scale	0.5887 ± 0.02	0.5910 ± 0.02
breastcancer	0.9565 ± 0.01	0.9561 ± 0.01
flags	0.2532 ± 0.02	0.2543 ± 0.03
german	0.6833 ± 0.03	0.6854 ± 0.03
wine	0.9470 ± 0.04	0.9493 ± 0.03
ilpd	0.5712 ± 0.04	0.5719 ± 0.04
ionosphere	0.9118 ± 0.03	0.9121 ± 0.03
knowledge	0.9224 ± 0.03	0.9233 ± 0.02
vertebral	0.8052 ± 0.05	0.8053 ± 0.04
sonar	0.7752 ± 0.05	0.7788 ± 0.05
skulls	0.2355 ± 0.06	0.2299 ± 0.06
diabetes	0.9685 ± 0.03	0.9701 ± 0.03
physio	0.8999 ± 0.03	0.9010 ± 0.03
breasttissue	0.6466 ± 0.07	0.6540 ± 0.08
ecoli	0.4971 ± 0.05	0.4977 ± 0.05
bupa	0.6888 ± 0.04	0.6930 ± 0.04
cleveland	0.2799 ± 0.04	0.2869 ± 0.04
haberman	0.5765 ± 0.05	0.5760 ± 0.05
hayes.roth	0.7363 ± 0.09	0.7420 ± 0.09
monks	0.9994 ± 0.00	0.9996 ± 0.00
newthyroid	0.4398 ± 0.04	0.4362 ± 0.04
yeast	0.4270 ± 0.03	0.4303 ± 0.03
spam	0.9224 ± 0.01	0.9262 ± 0.01
lymphography	0.3950 ± 0.03	0.3963 ± 0.03
movement_libras	0.6755 ± 0.06	0.6752 ± 0.06
SAheart	0.6388 ± 0.04	0.6376 ± 0.04
zoo	0.6886 ± 0.11	0.6948 ± 0.11

A large evaluation experiment was performed to demonstrate the ability of KalmanTune tuning to improve

TABLE 12. MAdaBoost and KalmanTune-M F-Scores with 10% label noise.

	MAdaBoost	KalmanTune-M
mushroom	0.9952 ± 0.00	0.9949 ± 0.00
iris	0.9449 ± 0.03	0.9442 ± 0.03
glass	0.5919 ± 0.10	0.5972 ± 0.09
car_eval	0.8491 ± 0.04	0.8501 ± 0.04
cmc	0.5264 ± 0.03	0.5273 ± 0.03
tvowel	0.8115 ± 0.03	0.8154 ± 0.03
dermatology	0.9408 ± 0.03	0.9422 ± 0.03
balance_scale	0.5865 ± 0.02	0.5885 ± 0.02
breastcancer	0.9557 ± 0.02	0.9558 ± 0.01
flags	0.2515 ± 0.02	0.2501 ± 0.02
german	0.6694 ± 0.03	0.6720 ± 0.03
wine	0.9433 ± 0.05	0.9416 ± 0.04
ilpd	0.5718 ± 0.04	0.5760 ± 0.04
ionosphere	0.9033 ± 0.04	0.9045 ± 0.04
knowledge	0.9227 ± 0.03	0.9232 ± 0.03
vertebral	0.8010 ± 0.05	0.8034 ± 0.05
sonar	0.7604 ± 0.06	0.7624 ± 0.05
skulls	0.2230 ± 0.06	0.2211 ± 0.07
diabetes	0.9603 ± 0.04	0.9598 ± 0.04
physio	0.9024 ± 0.03	0.9029 ± 0.03
breasttissue	0.6405 ± 0.09	0.6393 ± 0.09
ecoli	0.4992 ± 0.04	0.4996 ± 0.04
bupa	0.6815 ± 0.05	0.6811 ± 0.05
cleveland	0.2861 ± 0.05	0.2847 ± 0.05
haberman	0.5599 ± 0.06	0.5579 ± 0.06
hayes.roth	0.7158 ± 0.09	0.7143 ± 0.09
monks	0.9900 ± 0.02	0.9911 ± 0.02
newthyroid	0.4633 ± 0.04	0.4583 ± 0.04
yeast	0.4214 ± 0.03	0.4228 ± 0.03
spam	0.9203 ± 0.01	0.9214 ± 0.01
lymphography	0.3903 ± 0.03	0.3967 ± 0.04
movement_libras	0.6568 ± 0.06	0.6602 ± 0.06
SAheart	0.6315 ± 0.05	0.6300 ± 0.05
zoo	0.6828 ± 0.12	0.6873 ± 0.11

TABLE 13. MAdaBoost and KalmanTune-M F-Scores with 15% label noise.

	MAdaBoost	KalmanTune-M
mushroom	0.9939 ± 0.00	0.9935 ± 0.00
iris	0.9425 ± 0.04	0.9424 ± 0.04
glass	0.5597 ± 0.10	0.5601 ± 0.09
car_eval	0.8296 ± 0.04	0.8302 ± 0.04
cmc	0.5263 ± 0.03	0.5260 ± 0.03
tvowel	0.8032 ± 0.03	0.8057 ± 0.03
dermatology	0.9406 ± 0.03	0.9403 ± 0.03
balance_scale	0.5851 ± 0.02	0.5869 ± 0.03
breastcancer	0.9549 ± 0.02	0.9547 ± 0.02
flags	0.2570 ± 0.04	0.2603 ± 0.04
german	0.6681 ± 0.03	0.6696 ± 0.03
wine	0.9392 ± 0.05	0.9386 ± 0.05
ilpd	0.5868 ± 0.05	0.5878 ± 0.05
ionosphere	0.8981 ± 0.04	0.8977 ± 0.04
knowledge	0.9195 ± 0.03	0.9205 ± 0.03
vertebral	0.7925 ± 0.05	0.7887 ± 0.05
sonar	0.7461 ± 0.07	0.7475 ± 0.07
skulls	0.2208 ± 0.05	0.2196 ± 0.06
diabetes	0.9559 ± 0.03	0.9563 ± 0.03
physio	0.8997 ± 0.03	0.9003 ± 0.03
breasttissue	0.6167 ± 0.08	0.6196 ± 0.08
ecoli	0.4951 ± 0.04	0.4957 ± 0.04
bupa	0.6662 ± 0.05	0.6657 ± 0.05
cleveland	0.2853 ± 0.05	0.2882 ± 0.05
haberman	0.5564 ± 0.06	0.5590 ± 0.06
hayes.roth	0.6801 ± 0.08	0.6864 ± 0.08
monks	0.9783 ± 0.03	0.9803 ± 0.03
newthyroid	0.4672 ± 0.04	0.4660 ± 0.04
yeast	0.4174 ± 0.03	0.4192 ± 0.03
spam	0.9132 ± 0.01	0.9014 ± 0.01
lymphography	0.3895 ± 0.03	0.3930 ± 0.04
movement_libras	0.6579 ± 0.05	0.6606 ± 0.05
SAheart	0.6182 ± 0.05	0.6184 ± 0.04
zoo	0.6650 ± 0.11	0.6851 ± 0.12

the performance of ensemble models trained using the AdaBoost and MAdaBoost algorithms. Class-label noise was synthetically induced into datasets to demonstrate the ability of KalmanTune tuning to alleviate the impact of this. The results of this experiment show that, on average, KalmanTune tuning improves the performance of ensemble models trained using both AdaBoost and MAdaBoost. For models trained using AdaBoost, KalmanTune continued to improve performance more and more, as the amount of induced class-label noise was increased. Although it did lead to an improvement, as the amount of class-label noise in the training dataset increased, the impact of applying KalmanTune to models trained using MAdaBoost became less significant. This is evidence of the ability of MAdaBoost to handle some level of class-label noise.

In the future, it would be interesting to study the effects of the use of Kalman filter similarly on other ensembles. In some cases the Kalman filter can converge too quickly. This paper uses a trivial method, which can be made adaptive or further improved in other ways. Also, it would be interesting to explore the possibility of a linear time update step and multiple measurements to build a better model. Further comparisons with other noise sensitive and noise robust methods, like neural networks, other boosting methods, can be done and how KalmanTune affects on them can be further studied.

TABLE 14. MAdaBoost and KalmanTune-M F-Scores with 20% label noise.

	MAdaBoost	KalmanTune-M
mushroom	0.9922 ± 0.00	0.9919 ± 0.00
iris	0.9372 ± 0.04	0.9349 ± 0.04
glass	0.5545 ± 0.09	0.5559 ± 0.10
car_eval	0.8195 ± 0.05	0.8200 ± 0.05
cmc	0.5160 ± 0.03	0.5178 ± 0.03
tvowel	0.7973 ± 0.03	0.7983 ± 0.03
dermatology	0.9376 ± 0.03	0.9398 ± 0.03
balance_scale	0.5817 ± 0.02	0.5835 ± 0.03
breastcancer	0.9505 ± 0.02	0.9489 ± 0.02
flags	0.2618 ± 0.04	0.2621 ± 0.04
german	0.6549 ± 0.03	0.6557 ± 0.04
wine	0.9282 ± 0.05	0.9296 ± 0.05
ilpd	0.5852 ± 0.04	0.5814 ± 0.04
ionosphere	0.8877 ± 0.04	0.8837 ± 0.03
knowledge	0.9167 ± 0.03	0.9188 ± 0.03
vertebral	0.7765 ± 0.05	0.7740 ± 0.05
sonar	0.7363 ± 0.07	0.7386 ± 0.07
skulls	0.2233 ± 0.05	0.2245 ± 0.05
diabetes	0.9455 ± 0.05	0.9444 ± 0.05
physio	0.8950 ± 0.03	0.8938 ± 0.03
breasttissue	0.6171 ± 0.09	0.6196 ± 0.09
ecoli	0.5022 ± 0.05	0.5025 ± 0.06
bupa	0.6416 ± 0.05	0.6405 ± 0.06
cleveland	0.2792 ± 0.05	0.2799 ± 0.05
haberman	0.5690 ± 0.06	0.5696 ± 0.06
hayes.roth	0.6436 ± 0.09	0.6448 ± 0.09
monks	0.9379 ± 0.05	0.9416 ± 0.05
newthyroid	0.4771 ± 0.04	0.4745 ± 0.04
yeast	0.4136 ± 0.04	0.4146 ± 0.03
spam	0.8953 ± 0.01	0.8942 ± 0.01
lymphography	0.3985 ± 0.04	0.4025 ± 0.05
movement_libras	0.6257 ± 0.06	0.6261 ± 0.06
SAheart	0.6029 ± 0.05	0.6008 ± 0.05
zoo	0.6333 ± 0.12	0.6363 ± 0.12

APPENDIX

See Tables 5–14.

REFERENCES

[1] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[2] Y. Freund, "An adaptive version of the boost by majority algorithm," *Mach. Learn.*, vol. 43, no. 3, pp. 293–318, 2001.

[3] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Mach. Learn.*, vol. 40, no. 2, pp. 139–157, 2000.

[4] C. Domingo and O. Watanabe, "MadaBoost: A modification of AdaBoost," in *Proc. COLT*, 2000, pp. 180–189.

[5] P. S. Maybeck, *Stochastic Models, Estimation and Control*, vol. 3. New York, NY, USA: Academic, 1982.

[6] G. Bishop *et al.*, "An introduction to the Kalman filter," *Proc SIGGRAPH, Course*, vol. 8, nos. 27599–23175, p. 41, 2001.

[7] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class AdaBoost," *Statist. Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[8] J. Bootkrajang and A. Kaban, "Boosting in the presence of label noise," 2013, *arXiv:1309.6818*. [Online]. Available: http://arxiv.org/abs/1309.6818

[9] T. Kanamori, T. Takenouchi, S. Eguchi, and N. Murata, "Robust loss functions for boosting," *Neural Comput.*, vol. 19, no. 8, pp. 2183–2244, Aug. 2007.

[10] J. K. Bradley and R. E. Schapire, "Filterboost: Regression and classification on large datasets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 185–192.

[11] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.

[12] G. Rätsch, T. Onoda, and K.-R. Müller, "Regularizing AdaBoost," in *Proc. Neural Inf. Process. Syst. (NIPS)*, 1998, pp. 564–570.

[13] H. Masnadi-shirazi and N. Vasconcelos, "On the design of loss functions for classification: Theory, robustness to outliers, and SavageBoost," in *Proc. Neural Inf. Process. Syst. (NIPS)*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Red Hook, NY, USA: Curran Associates, 2009, pp. 1049–1056.

[14] R. Faragher, "Understanding the basis of the Kalman filter via a simple and intuitive derivation [lecture notes]," *IEEE Signal Process. Mag.*, vol. 29, no. 5, pp. 128–132, Sep. 2012.

[15] A. Pakrashi and B. B. Chaudhuri, "A Kalman filtering induced heuristic optimization based partitional data clustering," *Inf. Sci.*, vol. 369, pp. 704–717, Nov. 2016.

[16] R. Toscano and P. Lyonnet, "A new heuristic approach for non-convex optimization problems," *Inf. Sci.*, vol. 180, no. 10, pp. 1955–1966, May 2010.

[17] C. K. Monson and K. D. Seppi, "The Kalman swarm," in *Genetic and Evolutionary Computation—GECCO*, K. Deb, Ed. Berlin, Germany: Springer, 2004, pp. 140–150.

[18] A. Pakrashi and B. M. Namee, "Kalman filter-based heuristic ensemble (KFHE): A new perspective on multi-class ensemble classification using Kalman filters," *Inf. Sci.*, vol. 485, pp. 456–485, Jun. 2019.

[19] K. Yu, L. Wang, and Y. Yu, "Ordering-based Kalman filter selective ensemble for classification," *IEEE Access*, vol. 8, pp. 9715–9727, 2020.

[20] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences. [Online]. Available: http://archive.ics.uci.edu/ml

[21] L. Breiman and L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[22] J. D. Kelleher, B. M. Namee, and A. D'Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. Cambridge, MA, USA: MIT Press, 2015.

[23] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

ARJUN PAKRASHI received the B.Sc. degree (Hons.) in computer science from Calcutta University (CU), India, in 2011, the master's degree in computer science from Banaras Hindu University (BHU), India, in 2013, and the Ph.D. degree in computer science from University College Dublin, in 2020. He worked in the industry until 2015. He is currently a Postdoctoral Researcher with the Insight Centre for Data Analytics, University College Dublin. His research interests include machine learning, multi-label classification, ensemble methods, and anomaly detection.

BRIAN MAC NAMEE received the Ph.D. degree in computer science from Trinity College Dublin, Ireland, in 2004. After a period working in the industry and at the Dublin Institute of Technology, he joined University College Dublin, Ireland, in 2015, where he is currently an Associate Professor, the Director of the SFI Centre for Research Training in Machine Learning, and a Funded Investigator with the Insight and VistaMilk SFI research centres. He has coauthored the textbook *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples and Case Studies* which was published in 2015 with MIT Press. His research interests include machine learning, and in particular novelty detection, human-in-the-loop machine learning, data visualisation for machine learning, and machine learning applications.