Applied Intelligence
https://doi.org/10.1007/s10489-018-1341-9

# relf: robust regression extended with ensemble loss function

**Hamideh Hajiabadi**[1] · **Reza Monsefi**[1] · **Hadi Sadoghi Yazdi**[1]



**Abstract**
Ensemble techniques are powerful approaches that combine several weak learners to build a stronger one. As a meta-learning framework, ensemble techniques can easily be applied to many machine learning methods. Inspired by ensemble techniques, in this paper we propose an ensemble loss functions applied to a simple regressor. We then propose a half-quadratic learning algorithm in order to find the parameter of the regressor and the optimal weights associated with each loss function. Moreover, we show that our proposed loss function is robust in noisy environments. For a particular class of loss functions, we show that our proposed ensemble loss function is Bayes consistent and robust. Experimental evaluations on several data sets demonstrate that the our proposed ensemble loss function significantly improves the performance of a simple regressor in comparison with state-of-the-art methods.

**Keywords** Loss function · Ensemble methods · Bayes consistent loss function · Robustness

## 1 Introduction

Loss functions are fundamental components of machine learning systems and are used to train the parameters of the learner model. Since standard training methods aim to determine the parameters that minimize the average value of the loss given an annotated training set, loss functions are crucial for successful trainings [49, 55]. Bayesian estimators are obtained by minimizing the expected loss function. Different loss functions lead to different Optimum Bayes with possibly different characteristics. Thus, in each environment the choice of the underlying loss function is important, as it will impact the performance [44, 48].

Letting $\hat{\theta}$ denote the estimated parameter of a correct parameter $\theta$, the loss function $L(\hat{\theta}, \theta)$ is a positive function which assigns a loss value to each estimation, indicating how inaccurate the estimation is [42]. Loss functions assign a value to each sample, indicating how much that sample contributes to solving the optimization problem. Each loss function comes with its own advantages and disadvantages. In order to put our results in context, we start by reviewing three popular loss functions (0-1, Ramp and Sigmoid) and we will give an overview of their advantages and disadvantages.

Loss functions assign a value to each sample representing how much that sample contributes to solving the optimization problem. If an outlier is given a very large value by the loss function, it might dramatically affect the decision function [18]. The 0-1 loss function is known as a robust loss because it assigns value 1 to all misclassified samples — including outliers — and thus an outlier does not influence the decision function, leading to a robust learner. On the other hand, the 0-1 loss penalizes all misclassified samples equally with value 1, and since it does not enhance the margin, it cannot be an appropriate choice for applications with marginal importance [49].

The Ramp loss function, as another type of loss function, is defined similarly to the 0-1 loss function with the only difference that ramp loss functions also penalize some correct samples, those with small margins. This minor difference makes the Ramp loss function appropriate for applications with marginal importance [43, 49]. On the downside, the Ramp loss function is not differentiable, and hence not suitable for optimization purposes.

The Sigmoid loss function is almost the same as 0-1 loss functions, except that it is differentiable, which in turn makes optimization significantly easier. However, it

✉ Reza Monsefi
monsefi@um.ac.ir

Hamideh Hajiabadi
Hamideh.hajiabadi@mail.um.ac.ir

Hadi Sadoghi Yazdi
h-sadoghi@um.ac.ir

[1] Computer Department, Ferdowsi University of Mashhad (FUM), Mashhad, Iran

Published online: 12 November 2018

assigns a (very small) non-zero value to correct samples, meaning that those samples would also contribute to solving the optimization problem. Hence, in spite of easier optimization, the Sigmoid loss function leads to a less sparse optimization problem in comparison with the 0-1 loss function [45].

There are many other examples of different loss functions showing that while a loss function might be good for certain applications, it might be unsuitable for many others. Inspired by ensemble methods, in this paper we propose the use of an ensemble of loss functions during the training stage. The ensemble technique is one of the most influential learning approaches. Theoretically, it can boost weak learners, whose accuracies are slightly better than random guesses, into arbitrarily accurate strong learners [36]. This method would be effective when it is difficult to design a powerful learning algorithm directly [1, 30, 52]. As a meta-learning framework, it can be applied to almost all machine learning algorithms to improve their prediction accuracies.

Our goal in this paper is to propose a new ensemble loss function, which we later apply to a simple regressor. Half-Quadratic (HQ) minimization, which is a fast alternating direction method, is used to learn regressor's parameters. In each iteration, the HQ tries to approximate the convex or non-convex cost function with a convex one and pursue optimization [15]. Our main contributions are as follows.

- Inspired by ensemble-induced methods, we propose an ensemble loss whose properties are inherited from its base loss functions. Moreover, we show that each loss is a special case of our proposed loss function.
- We develop both online and offline learning frameworks to find the weights associated with each loss function, and so to build an ensemble loss function. For a particular class of base losses, we prove that the resulting ensemble loss function is Bayes consistent and robust.

This paper is structured as follows. We review some existing loss functions and several promising ensemble regressors in Section 2 which contains two subsections for each. We briefly explain about Half-Quadratic (HQ) programming in Section 3. Our proposed framework is discussed in Section 4, for which we provide implementation and test results in Section 5. Finally, we conclude in Section 6 with a list of problems for future work.

## 2 A review of loss functions

This work draws on two broad areas of research: loss functions, and ensemble-based regression methods. In this section, these two areas are fully covered.

### 2.1 A review of loss functions

In machine learning, loss functions are divided in two categories, margin-based and distance-based [42]. Margin-based loss functions are used for classification [2, 13, 24, 54], while Distance-based loss functions are generally used for regression. In this paper, we only focus on distance-based loss functions.

**Distance-Based Loss Functions** Let $\mathbf{x}$, $y$, $f(\mathbf{x})$ denote an input, the corresponding true label and the estimated label respectively. A distance-based loss function, is a penalty function $\phi(y - f(\mathbf{x}))$ where $y - f(\mathbf{x})$ is called distance [8, 40]. The risk associated with the loss function $\phi(.)$ described as

$$R_{\phi,P}(f) = \int_{X\times Y} \phi(y - f(\mathbf{x}))dP(X, Y)$$

where $P(X, Y)$ is the joint probability distribution over $X$ and $Y$. The ultimate aim of a learning algorithm is to find a function $f^*$ among a fixed class of function $\mathcal{F}$ for which the risk $R_{\phi,P}(f^*)$ is minimal [37, 42],

$$f^* = \arg\min_{f\in\mathcal{F}} R_{\phi,P}(f).$$

Generally $R(f)$ cannot be computed because the distribution $P(X, Y)$ is unknown. However, an approximation of $R(f)$ which is called empirical risk can be computed by averaging the loss function on the training set containing $n$ samples [20],

$$R_{\text{emp}}(f) = \frac{1}{n}\sum_{i=1}^{n} \phi(y_i - f(\mathbf{x_i})), \text{ and}$$

$$\hat{f} = \arg\min_{f\in\mathcal{F}} R_{\text{emp}}(f).$$

A loss function should be such that we arrive at Bayes decision function after minimizing the associated risk under given loss function. The loss is said to be Bayes consistent if by increasing the samples size, the resulting function converges to the Bayes decision function [6, 14, 53]. The Bayes decision function is fully explained in the cited papers.

Table 1 and Fig. 1 illustrate known examples of Bayes consistent loss functions [31, 32]. Figure 1 shows that all of these functions are convex and unbounded. As we

**Table 1** Well-known bayes consistent loss functions

| Algorithm | $\phi(y, f(\mathbf{x}))$ |
|---|---|
| Square [29] | $(1 - yf(\mathbf{x}))^2$ |
| Hinge [3] | $\max(0, 1 - yf(\mathbf{x}))$ |
| Exp. | $\exp(-\beta yf(\mathbf{x}))$ |
| Logistic [12] | $\ln(1 + e^{-(yf(\mathbf{x}))})$ |

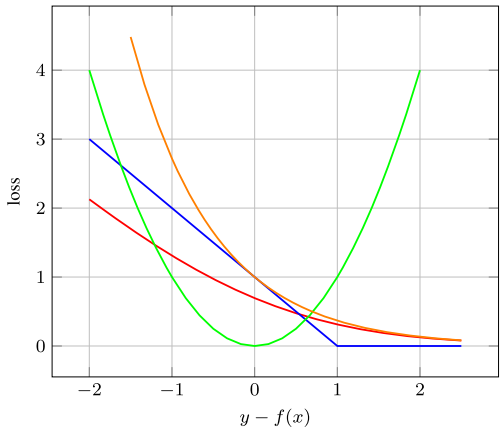



**Fig. 1** Well-known Bayes consistent loss functions, Hinge (blue), Square (green), Logistic (red) Exp. (orange)

mentioned in the previous section, loss functions assign a value to each sample which indicates how much that sample contributes to solving the optimization problem. Unbounded loss functions assign large values to samples with large errors and thus they are more sensitive to noise. Hence, under unbounded loss functions, the robustness deteriorates in noisy environments.

Figure 2 shows two unbounded loss functions (the Exp. loss and the Logistic loss) and a bounded one (the Savage loss). SavageBoost which uses the Savage loss function leads to a more robust learner in comparison with AdaBoost and Logitboost which uses the Exp. loss and the Logistic loss function respectively [32]. Several researchers suggested that although convex loss functions make optimization easier, the robustness deteriorates in the presence of outliers [35]. For example, while LS-SVR uses the Square loss and is sensitive to outliers, RLS-SVR uses the non-convex least squares loss function to overcome the limitation of LS-SVR [47].

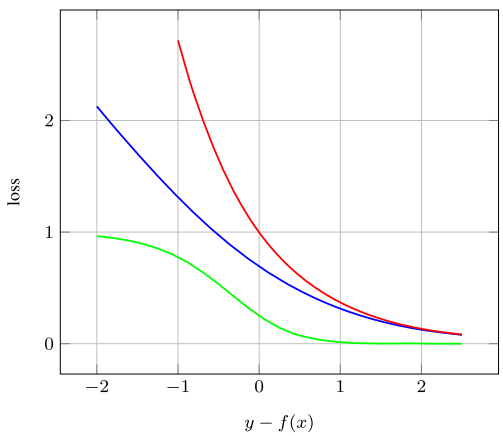



**Fig. 2** The Exp. loss (red), the Logistic loss (blue) and the Savage loss function (green)

There are many other distance-based loss functions which are not considered as Bayes consistent but have been widely used in literature. Some of these are shown in Table 2.

The Huber loss function which is the combination of the Square and the Absolute function is shown in Table 2. It was utilized in robust regression, and the results showed a significant improvement in its robustness comparing with the standard regression. The only difference between Absolute and Huber loss functions is at the point of $y - f(\mathbf{x}) = 0$. While the Absolute loss function is not differentiable at the point of $y - f(\mathbf{x}) = 0$, this problem is fully addressed by the Huber loss function [21].

Correntropy which is rooted from the Renyi's entropy is a local similarity measure. It is based on the probability of how much two random variables are similar in the neighbourhood of the joint space. The joint space can be adjusted by a kernel bandwidth. The C-loss function which is shown in Table 2 is inspired by Correntropy criteria and is known as a robust loss function [26]. Several researchers have used the C-loss function to improve the robustness of their learning algorithms [7, 9, 27, 38, 56].

$\epsilon$-insensitive Ramp-loss which is inspired by Ramp loss function [49] is proposed in [43]. It is a kind of robust and margin enhancing loss function which was applied to a linear and kernel Support Vector Regression (SVR) in [43]. The weights assigned to each sample are limited to be no higher than a pre-defined value of $\alpha$, thus the negative effect brought by outliers can be effectively reduced.

Loss functions can also be used in Dimensional Reduction (DR) purposes which is a kind of feature reduction technique. For example, in [50] nuclear norm (N norm) is used as the loss function in solving the DR problem based on matrix regression model. N norm can well preserve the low-rank information of samples and result in a low dimensional data and would be a good choice for DR purposes.

While each individual loss function has its own advantages and disadvantages, in this paper we propose an ensemble loss function which is a combination of several individual loss functions. By doing so, we hope to produce a strong ensemble loss function which its advantages are inherited from each individual loss function. In the next section, Ensemble learning and several promising ensemble methods are discussed.

## 2.2 Ensemble learning

Ensemble learning combines outputs of several models to make a prediction. They aim to improve the overall accuracy and robustness over individual models [11, 33]. The focus of most ensemble methods is on classification problems, however relatively few have paid attention to regression

**Table 2** Some widely used loss functions

| Name | Description |
|---|---|
| Absolute [46] | $l_{abs} = \|y - f(\mathbf{x})\|$ |
| Huber [21] | $l_{Huber} = \begin{cases} \frac{(y-f(\mathbf{x}))^2}{4\epsilon} & \text{for } \|y - f(\mathbf{x})\| < 2\epsilon \\ \|y - f(\mathbf{x})\| - \epsilon & \text{otherwise} \end{cases}$ |
| C-loss [26] | $l_{corr} = 1 - \exp(\frac{-(y-f(\mathbf{x}))^2}{2\sigma^2})$ |
| $\epsilon$-insensitive ramp-loss [43] | $l_{\epsilon-Ramp} = \begin{cases} \alpha & \text{for } \|y - f(\mathbf{x})\| \geq \alpha \\ \|y - f(\mathbf{x})\| & \text{for } \epsilon < \|y - f(\mathbf{x})\| < \alpha \\ 0 & \text{for } \|y - f(\mathbf{x})\| \leq \epsilon \end{cases}$ |

tasks. Ensemble methods are comprised of two steps: (1) generation step in which a set of base learners are built (2) integration step which involves combining these base learners in order to make the final prediction. Base learners can be combined statistically or dynamically. A static combination uses predefined combination rules for all instances while the dynamic one makes use of different rules for different instances [10, 25]. In the following the most promising Ensemble methods are discussed.

Two of the most appealing ensemble methods for regression trees are Bagging and random forest. They are the most commonly used algorithms due to their simplicity, consistency and accuracy [4, 5]. Breiman Random Forest approach first constructs a multitude of decision trees and output the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. The trees are later modified to incorporate randomness, a split used at each node to select randomly feature subset. moreover, the subset considered in one node is completely independent of the subset in the previous node.

There are some interesting ensemble methods on neural networks, one which is based on negative correlation, is called Evolutionary Ensembles with Negative Correlation Learning (EENCL) [28]. It randomly changes the weights of an existing neural network by using mutation. It also obtains ensemble size automatically. The importance of this approach is due to its theoretical foundations. Another interesting method on neural networks was presented in 2003 [22] which builds the base learners and the ensemble one simultaneously. Therefore, it saves time in comparing with the methods that first generate base learners and then combine them.

There are some other research based on local experts' which are specialized in local prediction, aim to achieve better prediction accuracy in comparison with globally best models. Recently, a Locally Linear Ensemble Regressor (LLER) has been proposed which first divides the original dataset into some locally linear regions by employing an EM procedure and then builds a linear model per each region in order to form the ensemble model [23].

In this paper, we dynamically combine several loss functions and the weights associated with each individual loss is obtained through the training phase using Half-Quadratic (HQ) optimization. In the following section, we provide an overview of HQ optimization.

## 3 Half-Quadratic optimization

Let $\phi_v(.)$ be a function of vector $v \in R^n$ and defined as $\phi(v) = \sum_{j=1}^{n} \phi_v(v_i)$ where $v_i$ is the $j$th entry of $v$. In machine learning problems, we often aim to minimize an optimization problem like

$$min \sum_{j=1}^{n} \phi(v_j) + J(v) \tag{1}$$

where usually $v$ is the learner's parameters and $\phi(.)$ is a loss function which can be convex or non-convex and $J(v)$ is a convex penalty function on $v$ which is optional and considered as the regularization term. According to Half-Quadratic (HQ) optimization [19], for a fixed $v_j$ , we have

$$\phi(v_j) = \min_{p_j} Q(v_j, p_j) + \psi(p_j) \tag{2}$$

where $\psi(.)$ is the convex conjugate function of $\phi(.)$ and $Q(v_j, p_j)$ is an HQ function which is modeled by the additive or the multiplicative form. The additive and the multiplicative forms for $Q(v_j, p_j)$ are respectively formulated as $(v_j - p_j)^2$ and $\frac{1}{2} p_j v_j^2$. Let $Q_v(v, \mathbf{p}) = \sum_{j=1}^{n} Q(v_j, p_j)$, the vector form of (2) is as follow

$$\phi_v(v) = \min_{\mathbf{p}} Q_v(v, \mathbf{p}) + \sum_{j=1}^{n} \psi(p_j). \tag{3}$$

By substituting the (3) for $\phi(v_j)$ in (1), the following cost function is obtained,

$$\min_{v} \{\phi_v(v) + J(v)\} = \min_{\mathbf{p}, v} Q_v(v, \mathbf{p}) + \sum_{j=1}^{n} \psi(p_j) + J(v). \tag{4}$$

Assuming the variable $v$ fixed, the optimal value for $\mathbf{p}$ is obtained by the minimization function $\delta(.)$ which is computed as

$$p_j = \delta(.) = \arg\min_{\mathbf{p}_j} \left\{ Q(v, \mathbf{p}) + \sum_{j=1}^{n} \psi(p_j) \right\}$$

and it is only related to $\phi(.)$ (some specific forms of $\phi(.)$ and the corresponding $\delta(.)$ is shown in Fig. 3). For each $v$ the value of $\delta(.)$ is such that

$$Q(v_j, \delta(v_j)) + \psi(\delta(v_j)) \leq Q(v_j, p_j) + \psi(p_j).$$

The optimization problem 4 is convex since it is the summation of three convex functions and is can be minimized alternating steps as follows

$$\mathbf{p}^{(k+1)} = \delta(v^{(k)})$$

$$v^{(k+1)} = \arg\min_{v} \left\{ Q(v, \mathbf{p}^{(k+1)}) + J(v) \right\}$$

where $k$ denotes the iteration number. At each iteration, the objective function is decreased until convergence. The HQ method is fully explained in [16, 19].

## 4 The proposed method

Let $(X, Y)$ denote all samples where $X = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\}$ are the inputs and $Y = \{y_1, y_2, \ldots, y_N\}$ are the labels. Let $w$ be the parameters of the regressor which estimates the predicted label by $\hat{y} = f(\mathbf{x}, \mathbf{w})$. Let $\{\phi_i(y, \hat{y})\}_{i=1}^{m}$ denote $m$ weak loss functions. We aim to find an optimal $\mathbf{w}$ and the best weights, $\{\lambda_1, \lambda_2, \ldots, \lambda_m\}$, associated with each loss function. We need to add a further constraint to avoid yielding near zero values for all $\lambda_i$ weights. Our proposed ensemble loss function is defined in (5).

$$L = \sum_{k=1}^{m} \lambda_k \phi_k(y, \hat{y}), \sum_{k=1}^{m} \lambda_k = 1, \lambda_k \geq 0 \tag{5}$$

Figure 4 shows the model of our proposed method with the bold box representing the novelty of this paper. In the training phase, the weights associated with $M$ loss functions are learned and our proposed ensemble loss function is formed. To ease computations we select each $\phi_i(y - f(\mathbf{x}))$ from M-estimator functions introduced in Fig. 3.

The expected risk of the proposed loss function is defined as

$$R(f) = E_X \left[ \sum_{k=1}^{m} \lambda_k \phi_k(y, \hat{y}) \right] = \sum_{k=1}^{m} \lambda_k E[\phi_k(y, f(\mathbf{x}))]$$
$$s.t. \sum_{k=1}^{m} \lambda_k = 1, \lambda_k \geq 0. \tag{6}$$

As known, under the Square loss, the Bayes estimator is the posterior mean and the Bayes estimator with respect to the Absolute loss is the posterior median. As shown in

| estimators | $\ell_1$-$\ell_2$ | Fair | log-cosh | Welsch($\sigma^2 = 0.5$) | Huber ($\lambda = 0.1$) |
|---|---|---|---|---|---|
| Potential function $\phi(t)$ | $\sqrt{\alpha + t^2} - 1$ | $\frac{\lvert t\rvert}{\alpha} - \log(1 + \frac{\lvert t\rvert}{\alpha})$ | $\log(\cosh(\alpha t))$ | $1 - \exp(-\frac{t^2}{\sigma^2})$ | $\begin{cases} t^2/2 & \lvert t\rvert \leq \lambda \\ \lambda\lvert t\rvert - \frac{\lambda^2}{2} & \lvert t\rvert > \lambda \end{cases}$ |
| Multiplica -tive form $\delta(t)$ | $1/\sqrt{\alpha + t^2}$ | $1/\alpha(\alpha + \lvert t\rvert)$ | $\alpha \frac{\tanh(\alpha t)}{t}$ | $\exp(-\frac{t^2}{\sigma^2})$ | $\begin{cases} 1 & \lvert t\rvert \leq \lambda \\ \frac{\lambda}{\lvert t\rvert} & \lvert t\rvert > \lambda \end{cases}$ |
| Additive form $\delta(t)$ | $t - \frac{t}{\sqrt{\alpha + t^2}}$ | $t - \frac{t}{\alpha(\alpha + \lvert t\rvert)}$ | $t - \alpha \tanh(\alpha t)$ | $t - t\exp(-\frac{t^2}{\sigma^2})$ | $\begin{cases} 0 & \lvert t\rvert \leq \lambda \\ t - \lambda sign(t) & \lvert t\rvert > \lambda \end{cases}$ |

**Fig. 3** Some estimators and the equivalent $\delta(.)$ for additive and multiplicative HQ forms [19]

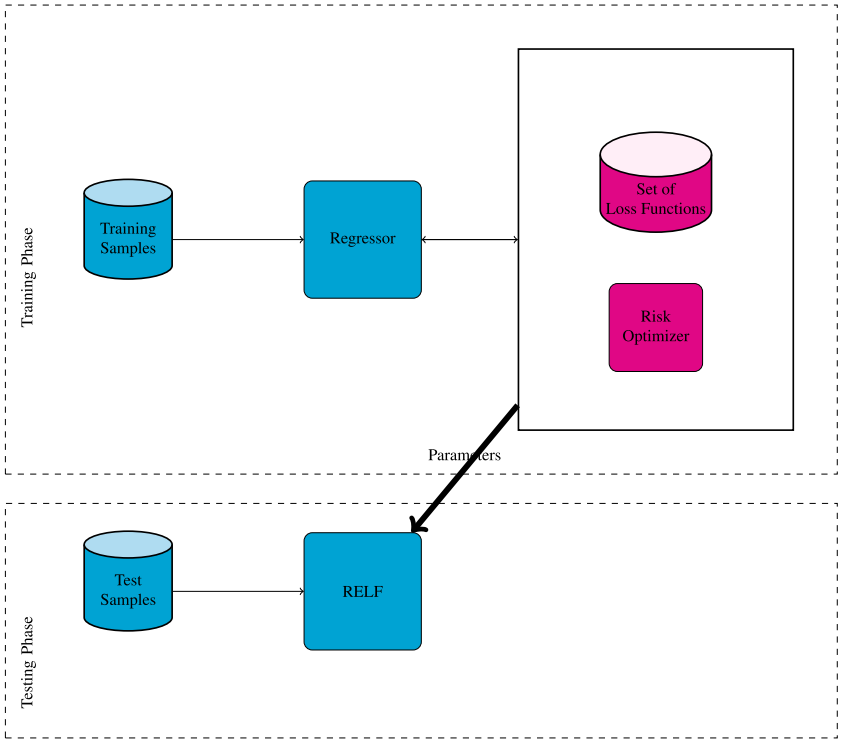



**Fig. 4** The proposed model

(6), the Bayes estimator of our proposed loss function is a weighted summation of all Bayes estimators associated with each individual loss function. For example, if we ensemble the Square and the Huber loss functions, then, the Bayes estimator of our ensemble loss function is a trade-off between the posterior mean and the median.

In the next two subsections, two properties of our proposed loss function are discussed. The third subsection contains the training scheme.

### 4.1 Bayes consistency

The advantages of using a Bayes-consistent loss function is fully explained in Section 2. In the following, it is proved that under some conditions, our proposed loss function is Bayes consistent.

**Theorem 1** *If each $\phi_k(y, f(\mathbf{x}))$ is a Bayes-consistent loss function, then $L_s = \sum_k \lambda_k \phi_k(y, f(\mathbf{x}))$ is also Bayes consistent.*

**Note** In binary classifications, it is known that the decision function $f^*$ is Bayes consistent if the following equation holds [2].

$$P(y=1|\mathbf{x}) > \frac{1}{2} \rightarrow f^*(\mathbf{x}) = 1,$$
$$P(y=1|\mathbf{x}) < \frac{1}{2} \rightarrow f^*(\mathbf{x}) = -1 \tag{7}$$

*Proof* Let $y$ be $\{+1, -1\}$, by assuming $\eta = P(Y=1|X)$, the conditional expected risk under each individual loss function, $\{\phi_k\}_{k=1}^m$, is written as (8) [32],

$$J_k = E[\phi(f(\mathbf{x}), Y)|X=\mathbf{x}] = \eta\phi_k(f(\mathbf{x}), 1) + (1-\eta)\phi_k(f(\mathbf{x}), -1). \tag{8}$$

Each loss function $\{\phi_k\}_{k=1}^m$ is convex because it is Bayes-consistent. Thus, the optimal decision function $f_k^*$, can be obtained by setting derivatives of $J_k$ to zero,

$$\frac{\partial J_k}{\partial f} = 0 \Rightarrow \eta\frac{\partial\phi_k(f(\mathbf{x}),1)}{\partial f(\mathbf{x})} + (1-\eta)\frac{\partial\phi_k(f(\mathbf{x}),-1)}{\partial f(\mathbf{x})} = 0 \Longrightarrow f_k^* \text{ is obtained}$$

The conditional expected risk for our proposed ensemble loss function is described as

$$c_{L(\eta)} = E[L_s(f(\mathbf{x}), y)|\mathbf{x}] = \eta\sum_k \lambda_k\phi_k(f(\mathbf{x}),1) + (1-\eta)\sum_k \lambda_k\phi_k(f(\mathbf{x}),-1)$$

Since $L_s(f(\mathbf{x}), y)$ is a linear combination of some convex functions, it is also convex. For example, by assuming $k=2$, the expected risk for the linear combination of two convex loss functions given an input $\mathbf{x}$ is written as

$$\begin{aligned} J &= c_{L(\eta)} = E[L(f(\mathbf{x}), y)|\mathbf{x}] \\ &= \lambda_1[\eta\phi_1(f(\mathbf{x}),1) + (1-\eta)\phi_1(f(\mathbf{x}),-1)] \\ &\quad +\lambda_2[\eta\phi_2(f(\mathbf{x}),1) + (1-\eta)\phi_2(f(\mathbf{x}),-1)] \\ &= \lambda_1 J_1 + \lambda_2 J_2 \end{aligned}$$

where $J_1, J_2$ have been shown in (8). The optimal decision function under our ensemble loss function is obtained by setting the derivative of $J$ to zero,

$$\frac{\partial J}{\partial f} = 0 \Rightarrow \lambda_1 \frac{\partial J_1}{\partial f(\mathbf{x})} + \lambda_2 \frac{\partial J_2}{\partial f(\mathbf{x})} = 0.$$

Figure 5 shows an example of two convex functions $J_1$ and $J_2$ with the optimal point $f_1^*$ and $f_2^*$ respectively. By looking at Fig. 5 the following equations are straightforward.

$$\begin{aligned} &\frac{\partial J_1}{\partial f(\mathbf{x})}|_{f_1^*} < 0 \quad \text{and} \quad \frac{\partial J_2}{\partial f(\mathbf{x})}|_{f_1^*} < 0 \Rightarrow \\ &\frac{\partial J}{\partial f}|_{f_1^*} = \lambda_1 \frac{\partial J_1}{\partial f(\mathbf{x})}|_{f_1^*} + \lambda_2 \frac{\partial J_2}{\partial f(\mathbf{x})}|_{f_1^*} < 0 \end{aligned} \tag{9}$$

$$\begin{aligned} &\frac{\partial J_1}{\partial f(\mathbf{x})}|_{f_2^*} > 0 \quad \text{and} \quad \frac{\partial J_2}{\partial f(\mathbf{x})}|_{f_2^*} > 0 \Rightarrow \\ &\frac{\partial J}{\partial f}|_{f_2^*} = \lambda_1 \frac{\partial J_1}{\partial f(\mathbf{x})}|_{f_2^*} + \lambda_2 \frac{\partial J_2}{\partial f(\mathbf{x})}|_{f_2^*} > 0 \end{aligned} \tag{10}$$

Having considered (9) and (10), we easily conclude that $f^*$ must lie between two points $f_1^*$ and $f_2^*$, and can be formulated as $f^* = \alpha_1 f_1^* + (1-\alpha_1) f_2^*$. It means that $f^*$ is a linear combination of $f_1^*, f_2^*$. Since for each $f_1^*, f_2^*$ (7) holds, it also holds for a linear combination of them. Therefore, $f^*$ is also Bayes consistent. This proof can easily be expanded for $k$ loss functions in the same way. □

### 4.2 Robustness

**Robust loss function** a distance-based loss function is said to be robust if there is a constant $k$ such that the loss function does not assign large values to samples with $e_i > k$ [17].

In the following, the mathematical explanation of the robustness is provided. Assuming a linear learner, the empirical risk is formulated as follows:

$$J = R_{emp} = \sum_{i=1}^{n} \phi(y_i - \mathbf{x}_i \cdot \mathbf{w}).$$

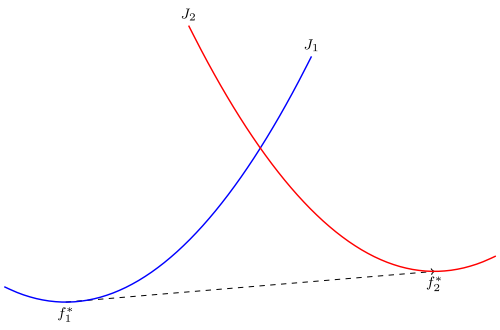



**Fig. 5** Expected Loss for Two Bayes-consistent Loss Function

where $\mathbf{x}_i \cdot \mathbf{w}$ denotes inner product of two vectors $\mathbf{x}$ and $\mathbf{w}$. To obtain the optimal value of $\mathbf{w}$, the following equation has to set to zero,

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{w}} = 0 \Rightarrow & \sum_{i=1}^{n} \frac{\partial \phi(y_i - \mathbf{x}_{i=1} \cdot \mathbf{w})}{\partial \mathbf{w}} \\ = & \sum_{i=1}^{n} \phi'(y_i - \mathbf{x}_i \cdot \mathbf{w}) \mathbf{x}_i = 0 \end{aligned}$$

where $e_i = y_i - \mathbf{x}_i \cdot \mathbf{w}$ . Given the weight function $w(e_i) = \frac{\phi'(e_i)}{e_i}$ , the above equation can be reformulated as follows:-

$$\sum_{i=1}^{n} \phi'(y_i - \mathbf{x}_i \cdot \mathbf{w}) x_i = \sum_{i=1}^{n} w(e_i) \mathbf{x}_i. \tag{11}$$

A loss function is robust if the following equation holds [17].

$$\exists k, \quad \forall \mathbf{x}_i, \ |e_i| > k \Longrightarrow w(e_i) \to 0. \tag{12}$$

**Theorem 2** *If all individual loss functions are robust, our proposed ensemble loss function is also robust.*

*Proof* For the ensemble loss function, $w(e_i)$ is written as follows:

$$w(e_i) = \sum_{k=1}^{m} \lambda_k w_k(e_i), \sum_{k=1}^{m} \lambda_k = 1, \lambda_k > 0$$

To prove Theorem 2 we need to prove that $\exists k, \ \forall \mathbf{x}_i \ | \ |e_i| > k \Longrightarrow \sum_{k=1}^{m} \lambda_k w_k(e_i) \to 0$. Since each individual loss function is robust then the following equation is straightforward.

$$\exists k_k, \quad \forall \mathbf{x}_i, \quad |e_i| > k_k \Longrightarrow w_k(e_i), \to 0$$

where $w_k(.)$ is the weight function corresponding to $\phi_k(.)$. By assuming $k = \max\{k_k\}_{k=1}^{m}$, the proof of (12) is straightforward. □

### 4.3 Training phase using our proposed loss function

We aim to minimize the empirical risk of our proposed loss function as follows

$$\begin{aligned} \min_{\mathbf{w},\lambda} R_{emp} = & \sum_{i=1}^{n} L_s(y_i - \mathbf{x}_i \cdot \mathbf{w}) = \sum_{i=1}^{n} \sum_{k=1}^{m} \lambda_k \phi_{ik}(y_i - \mathbf{x}_i \cdot \mathbf{w}) \\ = & \sum_{i=1}^{n} \sum_{k=1}^{m} \lambda_k \phi_{ik}(e_i) \end{aligned} \tag{13}$$

where $L_s(.)$ denotes the value of ensemble loss function for $i$th sample, $\lambda_k$ denotes the weights associated with the $k$th loss function and $\phi_{ik}$ denotes the value of the $k$th loss

function for the $i$th sample. We first omit $\lambda_k$ from (13) and will later show that the weights associated with each loss function appear implicitly through HQ optimization. According to HQ optimization, (13) is restated as follows

$$\min_{\mathbf{w},P} J = \sum_{i=1}^{n}\sum_{k=1}^{m} Q(P, e_i) + \psi_k(P)$$

where $\psi_k$ is the point-wise maximum of some convex functions and consequently it is convex, $e_i = y - \mathbf{w}\cdot\mathbf{x}_i$ is the error associated with the $i$th sample and $P$ is the auxiliary variable. By substituting $Q(P, e_i)$ with the multiplicative HQ function, the following equation is obtained.

$$\begin{aligned}\min_{\mathbf{w},P} J(\mathbf{w}, P) &= \sum_{i=1}^{n}\sum_{k=1}^{m} \frac{1}{2} p_{ik}(e_i)^2 + \psi_k(P) \\ &= \sum_{i=1}^{n}\sum_{k=1}^{m} \frac{1}{2} p_{ik}(y - \mathbf{w}\cdot\mathbf{x}_i)^2 + \psi_k(P) \quad (14)\end{aligned}$$

where $P$ is a matrix of $n$ rows and $m$ columns. And $p_{ik}$ denotes the element of $i$th row and $k$th column. To solve the above problem, we make use of HQ optimization algorithm which is iterative and alternating. Each iteration $(s)$ consists of two steps as follows [19]

**First** by considering $\mathbf{w} = \mathbf{w}^{(s)}$ constant, we arrive at the following optimization problem with one variable $P$

$$P^{(s+1)} = \arg\min_{P} \sum_{i=1}^{n}\sum_{k=1}^{m} \frac{1}{2} p_{ik}(y_i - \mathbf{w}^{(s)}\cdot\mathbf{x}_i)^2 + \psi_k(P). \quad (15)$$

The value of $P^{(s+1)}$ is calculated using $\delta(.)$ function which is pre-calculated for some special loss functions $\phi(.)$ and their corresponding conjugate functions $\psi(.)$ which are listed in Fig. 3. The optimal value of $p_{ik}$ is obtained by $\delta_k(e_i)$.

**Second** given the obtained value of $P^{(s+1)}$, we optimize over $\mathbf{w}$ as follows

$$\begin{aligned}\mathbf{w}^{(s+1)} &= \arg\min_{\mathbf{w}} j = \arg\min_{\mathbf{w}} \sum_{i=1}^{n}\sum_{k=1}^{m} \frac{1}{2} p_{ik}^{(s+1)}(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2 \\ \frac{\partial j}{\partial \mathbf{w}} &= 0 \Rightarrow \mathbf{w}^{(s+1)} \\ &= \left[\underbrace{\sum_{i=1}^{n} \mathbf{x}_i \times \mathbf{x}_i^T \times \sum_{k=1}^{m} p_{ik}^{(s+1)}}_{*}\right]^{-1} \sum_{i=1}^{n} \mathbf{x}_i \sum_{k=1}^{m} p_{ik}^{(s+1)} y_i \quad (16)\end{aligned}$$

where $P^{(s)}$ and $\mathbf{w}^{(s)}$ are respectively the optimal values for matrix $P$ and vector $\mathbf{w}$ in the $s$th iteration. The matrix represented by $*$ is generally Positive Semi-Definite (PSD) and might not be invertible. To make it Positive Definite (PD) we add it a diagonal matrix with strictly positive diagonal entries [34]. The refinement of (16) is as follows

$$\mathbf{w}^{(s+1)} = \left[\underbrace{\sum_{i=1}^{n} \mathbf{x}_i \times \mathbf{x}_i^T \times \sum_{k=1}^{m} p_{ik}^{(s+1)}}_{*} + \alpha \times I\right]^{-1} \sum_{i=1}^{n} \mathbf{x}_i \sum_{k=1}^{m} p_{ik}^{(s+1)} y_i \quad (17)$$

where $\alpha \in R$ is a very small positive number and $I$ is the identity matrix. The (15) and (17) are performed iteratively in an alternating manner until convergence.

$P^{(s+1)}$ denotes matrix $P$ in $s$th iteration. $p_{ik}^{(s)}$ is the element of the $i$th row and the $k$th column of matrix $P$, so, $p_{ik}$ can be considered as the weight of the $i$th sample associated with the $k$th loss function. Thus, we can calculate the total weight of the $k$th loss function by $\lambda_k = \sum_{i=1}^{n} p_{ik}$.

**Proposition 1** *The sequence* $\{J(P^{(s)}, \mathbf{w}^{(s)}), s = 1, 2, \ldots\}$ *generated by Algorithm 1 converges.*

*Proof* According to (15) and the properties of the minimizer function $\delta(.)$ we have

$$J(\mathbf{w}^{(s)}, P^{(s+1)}) \leq J(\mathbf{w}^{(s)}, P^{(s)})$$

for a fixed $\mathbf{w}^{(s)}$. And by (17), for a fixed $p^{(s+1)}$ we have

$$J(\mathbf{w}^{(s+1)}, P^{(s+1)}) \leq J(\mathbf{w}^{(s)}, P^{(s+1)}).$$

Summing up the above equations gives us:

$$J(\mathbf{w}^{(s+1)}, P^{(s+1)}) \leq J(\mathbf{w}^{(s)}, P^{(s+1)}) \leq J(\mathbf{w}^{(s)}, P^{(s)}).$$

Since the cost function $J(P, \mathbf{w})$ is below bounded, the sequence

$$\left\{\ldots (\mathbf{w}^{(s+1)}, P^{(s+1)}) \leq J(\mathbf{w}^{(s)}, P^{(s+1)}) \leq J(\mathbf{w}^{(s)}, P^{(s)}) \ldots\right\}$$

decrease and finally converges when $s \to \infty$ [19]. □

### 4.4 Toy example

To clarify our proposed approach, we carry out an experiment on small synthetic data. $x$ as input are generated from $[-20 : 20]$ by step size 0.5. Labels lie in a line defined by $y = 2 \times x + z$, where $z$ denotes noise. We ensemble two loss functions, Welsch and $l_1 l_2$, which are selected from functions listed in Fig. 3. We conduct our experiments in presence of two kinds of noises: (1) Zero-mean Gaussian noise and (2) Outliers.

The weights associated with each base loss function and $w$ are iteratively updated according to (15) and (17) respectively. These weights are presented in Table 3. The weights associated with Welsch loss function decreases in presence of outliers while it increases for the $l_1 l_2$ function. Welsch and the $l_1 l_2$ functions are plotted in Fig. 6.

**Algorithm 1** Our Proposed Ensemble Loss Function using HQ Optimization Algorithm (RELF)

**INPUT:**
Streaming samples $\{\mathbf{x}_i, y_i\}_{i=1}^n$
Base loss functions $\{\phi_i(\mathbf{x}_i, y_i)\}_{i=1}^m$
**OUTPUT:**
Parameter $\mathbf{w}$, $p$ in (14) and $\{\lambda_i\}_{i=1}^m$ (weights associated with each base loss function $\phi_i(.)$)

1: Initiate $L_s$ using $\{\phi_i(\mathbf{x}_i.y_i)\}_{i=1}^m$ and random $\lambda_i$
2: Initialized parameters $\mathbf{w} \sim N(0.\Sigma)$
3: **while** until convergence **do**
4: Compute $\{\phi_{ik}\}_{k=1}^m$ for $(\mathbf{x}_i.y_i)$
5: Update $P$ by $p_{ik} = \delta(\mathbf{x}_i, y_i)$ according to (15)
6: Update learning parameter $\mathbf{w}$ according to (17)
7: $s \leftarrow s + 1$
8: $\lambda_j = \sum_{i=1}^n p_{ij}$
9: **return** $\{P^{(s)}, \mathbf{w}^{(s)}, \lambda_i{}_{i=1}^n\}$

As shown in Fig. 6, Welsch is a bounded function and the $l_1 l_2$ is unbounded. Therefore, the $l_1 l_2$ assigns a larger value to a sample with large error in comparison with Welsch. Therefore, to decrease the influence of outlier on the predicted function, the value of $\lambda_i$ for $l_1 l_2$ is less than the corresponding value for Welsch function in presence of outliers.

**Table 3** Results for syntactic data

| $\lambda_i$ | Gaussian noise | Outliers |
|---|---|---|
| $\lambda_{Welsch}$ | 0.1546 | 0.3578 |
| $\lambda_{l_1 l_2}$ | 0.8453 | 0.6422 |
| $w$ | 1.9975 | 1.9969 |

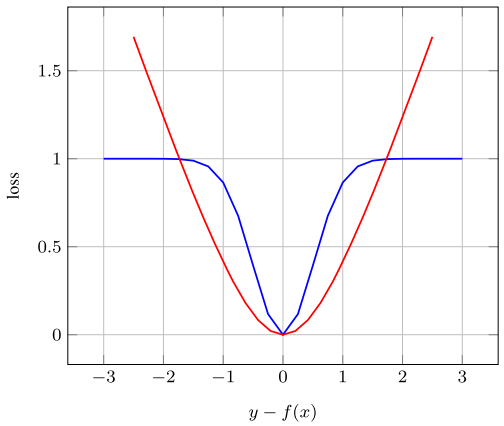



**Fig. 6** Welsch function (red line), $l_1 l_2$ function(red line)

We also combine three, four and five base loss functions in Fig. 3 and the weights associated with each loss function are reported in Table 4. Results show that among these five loss functions, Fair loss function contributes the most to form the ensemble loss function. Also, the predicted $w$ has been above value 1.99 for all experiments.

## 5 Experiments

In this section, we evaluate how our proposed loss function works in regression problems. To make our proposed ensemble loss function stronger, we have chosen base loss functions which are diversified by different behaviour against outliers. Welsch function is bounded and assigns a small value to samples with large errors. Huber loss function penalizes samples linearly with respect to $e_i = y - \mathbf{x}_i \cdot \mathbf{w}$, while $l_1 l_2$ function highly penalizes samples with large $e_i$. Therefore, these three loss functions which are completely diversified by the behaviours against outliers have been chosen as the base functions of our proposed ensemble loss function.

We have conducted our experiments on some different benchmark datasets which are briefly described in Table 5. Mean-Absolute Error (MAE) is utilized to compare the results which is calculated as $\frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$ where $N$ is the number of test samples and $y_i, \hat{y}_i$ are the true and estimated label respectively. In all experiments, we have used 10 fold cross validation for model selection. It means that the original dataset is partitioned into 10 disjoint subsets. Afterwards, we have run 10 iterations in each 9 subsets have been used for training and the remaining one for testing. Every subset has been exactly used once for testing. And the best model parameters has been selected.

The data have been initially normalized. Table 5 shows the experimental results on some benchmark datasets in natural situation (without adding outliers to samples). It

**Table 4** Weights associated with each base loss function

| Number of base loss functions | kind of noise | Welsch | $l_1$_$l_2$ | Huber | Fair | Log-cosh |
|---|---|---|---|---|---|---|
| 3 | Gaussian | 0.26 | 0.58 | 0.14 | – | – |
| 3 | Outlier | 0.28 | 0.62 | 0.09 | – | – |
| 4 | Gaussian | 0.15 | 0.36 | 0.07 | 0.38 | – |
| 4 | Outlier | 0.16 | 0.37 | 0.05 | 0.41 | – |
| 5 | Gaussian | 0.13 | 0.28 | 0.05 | 0.30 | 0.21 |
| 5 | Outlier | 0.11 | 0.27 | 0.04 | 0.33 | 0.23 |

**Table 5** The Mean Absolute Error (MAE) comparison results on several datasets. The best result for each dataset is presented in bold

| Dataset name | # of samples | # of features | Lasso | LARS | SVR | RELF |
|---|---|---|---|---|---|---|
| Airfoil self-noise | 1503 | 6 | 3.5076 | 3.5147 | 3.7954 | **3.4937** |
| Energy efficient dataset | 768 | 8 | 1.9741 | 1.9789 | **1**.4021 | 1.9476 |
| Red wine quality dataset | 4898 | 12 | 2.6605 | 2.6637 | 2.7630 | **2.6561** |
| White wine quality dataset | 4898 | 12 | 1.9741 | 1.9789 | 1.4021 | **1.9733** |
| Abalone dataset | 4177 | 8 | 2.0345 | 1.8693 | 1.5921 | **1.5549** |
| Bodyfat_Dataset | 252 | 13 | 0.91823 | 0.84867 | 0.6599 | **0.52465** |
| Building_Dataset | 4208 | 14 | 1.017 | 1.0183 | 2.4750 | **1.0024** |
| Engine_Dataset | 1199 | 2 | 1.3102 | 1.4113 | 0.9833 | **0.88694** |
| Vinyl_Dataset | 506 | 13 | 1.1501 | 1.0598 | 0.9854 | **0.23952** |
| Simplefit_Dataset | 94 | 1 | 0.63349 | 0.51819 | 0.4836 | **0.38523** |

**Table 6** Values of Cost Function, $J(\mathbf{w}, p)$ in several successive Iterations

| Dataset name | 1 | 2 | ... | 10 | ... | 29 | 30 | Decrease ratio | CPU time |
|---|---|---|---|---|---|---|---|---|---|
| Airfoil self-noise | 617.9089 | 615.9067 | .. | 615.2434 | .. | 615.2074 | 615.2070 | 0.986528 | 0.3125 |
| Energy efficient | 298.9774 | 294.7328 | .. | 291.9451 | .. | 291.9144 | 291.8603 | 0.988085 | 0.6094 |
| Red wine quality | 658.9971 | 657.8419 | .. | 657.5159 | .. | 657.4989 | 657.4986 | 0.988455 | 0.2656 |
| White wine quality | 324.2221 | 291.8042 | .. | 291.8410 | .. | 291.6763 | 291.6837 | 0.995166 | 1 |
| Abalone | 1.5604 | 1.5490 | .. | 1.5362 | .. | 1.5331 | 1.5330 | 0.883212 | 0.6250 |
| Bodyfat | 98.6765 | 98.4771 | .. | 98.4393 | .. | 98.4389 | 98.4389 | 0.998316 | 0.0469 |
| Building | 3.3932 | 3.3650 | .. | 3.3434 | .. | 3.3400 | 3.3399 | 0.934334 | 0.8438 |
| Engine | 1.6640 | 1.6026 | .. | 1.5500 | .. | 1.5397 | 1.5394 | 0.914928 | 0.5313 |
| Vinyl | 2.5582 | 2.5574 | .. | 2.5573 | .. | 2.5573 | 2.5573 | 1 | 1.9063 |
| Simplefit | 44.2226 | 43.4881 | .. | 43.1884 | .. | 43.1652 | 43.1649 | 0.977782 | 0.0156 |

**Table 7** Increase Ratio of MAE in the Face of Outliers (10%). The best result for each dataset is presented in bold

| Dataset name | Lasso | LARS | SVR | RELF |
|---|---|---|---|---|
| Airfoil self-noise | 1.0127 | 1.0191 | 1.0227 | **1.0016** |
| Energy efficient dataset | 1.4025 | 1.3641 | 1.0971 | **1.0082** |
| Red wine quality dataset | 1.0683 | 1.0544 | 1.0235 | **1.0139** |
| White wine quality dataset | 1.2941 | 1.2897 | 1.0971 | **1.0602** |
| Abalone dataset | 1.0264 | 1.1246 | 1.0183 | **1.0077** |
| Bodyfat_Dataset | 1.5125 | 1.5279 | **1.0242** | 1.0692 |
| Building_Dataset | 1.0016 | 1.0014 | 1.0143 | **1.0004** |
| Engine_Dataset | 1.1374 | 1.0242 | 1.0569 | **1.0089** |
| Vinyl_Dataset | 1.2345 | 1.148 | 1.1987 | **1.0978** |
| Simplefit_Dataset | 1.3122 | 1.8628 | **1.0023** | 1.2359 |

**Table 8** Increase ratio of MAE in the face of outliers (30%). The best result for each dataset is presented in bold

| Dataset name | Lasso | LARS | SVR | RELF |
|---|---|---|---|---|
| Airfoil self-noise | 5.9316 | 6.6159 | 1.0213 | **1.0178** |
| Energy efficient dataset | 2.544 | 2.0297 | 1.1555 | **1.0016** |
| Red wine quality dataset | 1.408 | 2.0033 | 1.0270 | **1.0024** |
| White wine quality dataset | 2.1278 | 2.0297 | 1.1555 | **1.0219** |
| Abalone dataset | 2.0264 | 3.0233 | 1.0338 | **1.0318** |
| Bodyfat_Dataset | 2.3491 | 1.1265 | 1.0338 | **1.2882** |
| Building_Dataset | 1.985 | 1.1438 | 1.0151 | **1.0022** |
| Engine_Dataset | 2.1156 | 2.0233 | 1.3665 | **1.0061** |
| Vinyl_Dataset | 3.5624 | 2.2455 | 1.2785 | **1.0651** |
| Simplefit_Dataset | 8.674 | 8.3125 | **1.4424** | 1.5212 |

**Table 9** Datasets' description

| Dataset | Source | Number of data | Number of feature |
|---|---|---|---|
| BodyFat | Source: StatLib / bodyfat | 252 | 14 |
| Abalone | Source: UCI / Abalone | 4,177 | 8 |
| Cadata | Source: StatLib / houses.zip | 20,640 | 8 |
| Cpusmall | Source: Delve / comp-activ | 8,192 | 12 |
| Housing | Source: UCI / Housing (Boston) | 506 | 13 |
| Mg | Source: [GWF01a] | 1,385 | 6 |

shows that the performance of all regressors are somehow the same in all datasets; however, RELF achieves the best results.

### 5.1 Discussion about the convergence of HQ optimization method

In this section, the convergence of HQ optimization method in our algorithm is experimentally studied. Table 6 shows values of cost function, $J(\mathbf{w}, P)$, in 30 successive iterations. We define a *decrease ratio* in (18) which shows how much cost function has been reduced in iteration 10 to 30. This ratio has been calculated for each dataset separately and shown in Table 6. The results shows that the HQ optimization method quickly converges within 30 iterations in all cases. Moreover, $J(\mathbf{w}^{(10)}, P^{(10)})$ is a good approximation of the optimal point. We have also provided the CPU time in seconds for each dataset in Table 6.

$$decrease\ \ ratio = \frac{J(\mathbf{w}^{(1)}, P^{(1)}) - J(\mathbf{w}^{(10)}, P^{(10)})}{J(\mathbf{w}^{(1)}, P^{(1)}) - J(\mathbf{w}^{(30)}, P^{(30)})} \quad (18)$$

### 5.2 Discussion about robustness

In this section, we investigate the robustness of our proposed loss function through the experiment on some benchmark datasets which are provided in Table 5. 10 fold cross validation has been used to tune the hyper-parameters. To study the robustness, we add outliers to training and validation samples. We conduct experiments on data which are corrupted with various level of outlier including 10% and 30%. 30% outliers which means that we randomly select 30% of samples and add outliers to their labels. Tables 7 and 8 show the increase ratio of MAE in the face of outliers in comparison of natural situation (without outliers). The best results are presented with bold. Lasso, LARS, and SVR are three well-known regressions which have been used to make comparison.

Table 7 lists the ratio of MAE value for data with 10% outliers to MAE value for original data. Except two datasets RELF has been least influenced by outliers among other regressors. Table 8 presents the numerical comparison results for the four mentioned models in the presence of 30% outliers. RELF was least influenced by outliers and SVR got the better results in comparison with LASSO and LARS.

### 5.3 Comparison with State-of-the-Art ensemble regressors

We also investigate the effectiveness of our proposed method in comparison with several promising ensemble regressors through experiments on some benchmark datasets. The datasets have been selected from LIBSVM data page.[1] We compare RELF with four ensemble regression methods: Locally Weighted Linear Regression (LWLR) [51], Locally Linear Ensemble for Regression (LLER) [23], Artificial Neural Network (ANN) [39] and Random Forest (RF) [39]. The first two models are based on combining local experts and the next two models are nonlinear. In the following, the implementation settings for each method is fully provided.

The LWLR which is based on local expert method aims to build a local linear regressor for each test sample based on its neighbours. The training of each linear model is such that it assigns higher weights to those training samples which are closer to the given test sample. The weights are calculated according to Gaussian kernel which is fully explained in [41].

For the ANN, the number of hidden nodes are selected from {1, 2, 3, 5, 10, 15, 20, 25, 30} set. If the amount of loss for validation batch fails to decrease during six successive epochs, the training phase ends.

To implement RF, we set the number of trees to 100 and bootstrap sample size to 80 percent of training samples for each tree. We set the minimum size of leaf nodes to {0.1, 0.2, 0.5, 1, 2, 5} percent of training samples.

In all experiments, we use 10 fold cross validation for parameter tuning, and the data are initially normalized into the range [-1, 1]. Root-Mean-Square-Error (RMSE) is utilized to compare the results which is calculated as $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$ where $N$ is the number of test samples and $y_i, \hat{y}_i$ are the true and estimated label respectively. Table 9 provides some extra information about each dataset.

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

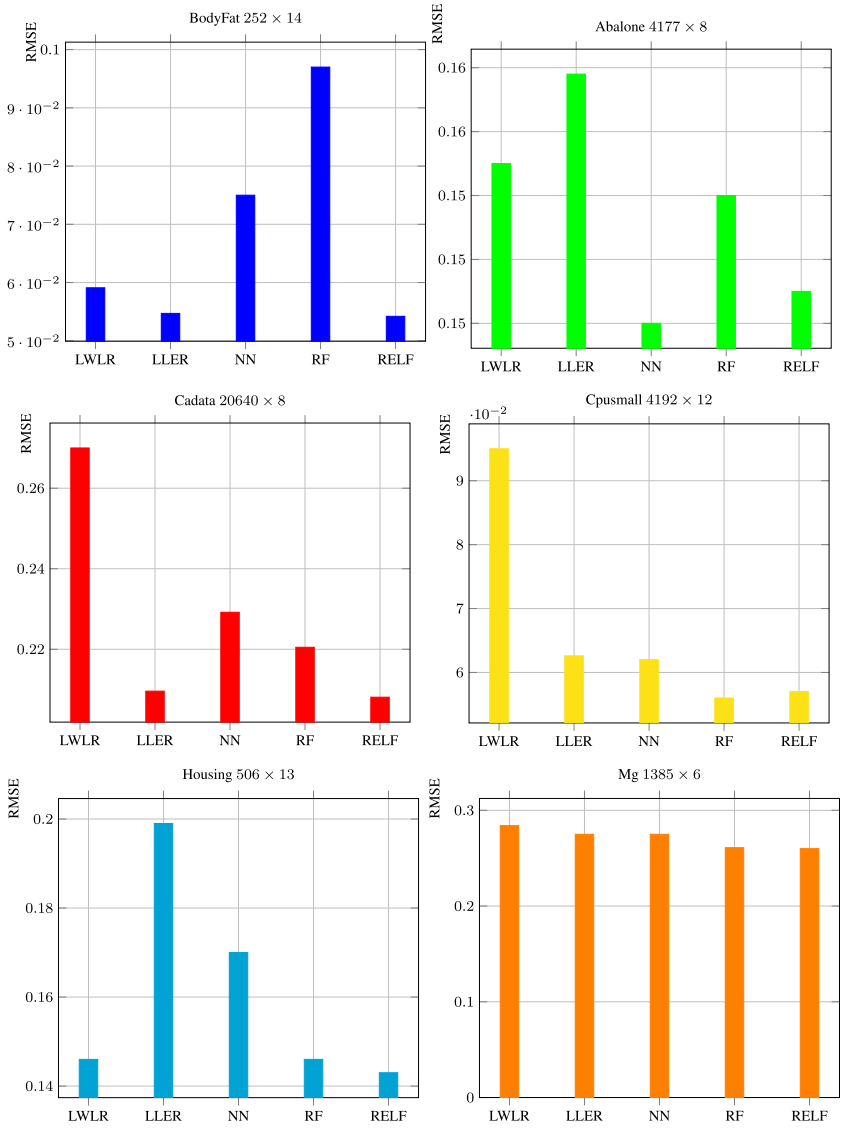



**Fig. 7** RMSE comparison results on benchmark datasets

The five mentioned models have been numerically compared on 6 datasets and the results are plotted by bar charts in Fig. 7. NN and RF models are nonlinear while LWLR and LLER are locally linear. The RELF yields the lowest RMSE value for almost all datasets except for Abalone and Cpusmall that in which NN and RF get the lowest RMSE value.

## 6 Conclusion and future works

Inspired by ensemble methods, in this paper we proposed a regression extended with an ensemble loss function which is called RELF. We focused on regression tasks and it is considered as an initial attempt to explore the use of ensemble loss functions. In this work, we utilized Half-Quadratic programming to find the optimal weight associated with each loss function.

RELF has been implemented and evaluated in noisy environments which showed a significant improvement with respect to outliers in comparison with simple regressors. We have also investigated the performance of RELF in comparison with some promising ensemble regressors through experiments on several benchmark datasets. Moreover, according to our results, the HQ optimization method has quickly converged.

As a future work, we aim to use an evolving solution that all the loss functions are applied and ranked on non-dominance. All those that are not dominated become the leaders. It would converge faster than individually applied loss functions. As another future work, we plan to make our proposed ensemble loss function sparse by adding

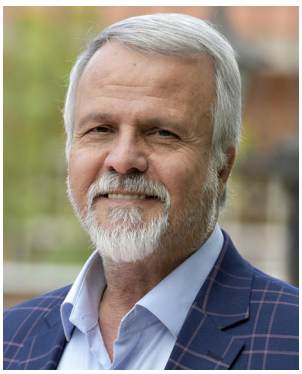
a regularization term. In addition, although the focus of this paper is on regression, the concept of ensemble loss function can also be applied to other classification based applications. Another possible extension is making the ensemble loss function sparse by adding a regularization term.

## References


1. Bai Q, Lam H, Sclaroff S (2014) A bayesian framework for online classifier ensemble. In: International conference on machine learning, pp 1584–1592
2. Bartlett PL, Jordan MI, McAuliffe JD (2006) Convexity, classification, and risk bounds. J Am Stat Assoc 101(473):138–156
3. Bartlett PL, Wegkamp MH (2008) Classification with a reject option using a hinge loss. J Mach Learn Res 9:1823–1840
4. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
5. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
6. Buja A, Stuetzle W, Shen Y (2005) Loss functions for binary class probability estimation and classification: structure and applications. Manuscript, available at www-stat.wharton.upenn.edu/∼buja
7. Chen B, Xing L, Liang J, Zheng N, Principe JC (2014) Steady-state mean-square error analysis for adaptive filtering under the maximum correntropy criterion. IEEE Signal Process Lett 21(7):880–884
8. Chen B, Xing L, Xu B, Zhao H, Zheng N, Principe JC (2017) Kernel risk-sensitive loss: definition, properties and application to robust adaptive filtering. IEEE Trans Signal Process 65(11):2888–2901
9. Chen B, Xing L, Zhao H, Zheng N, Prı JC et al (2016) Generalized correntropy for robust adaptive filtering. IEEE Trans Signal Process 64(13):3376–3387
10. Cruz RM, Sabourin R, Cavalcanti GD, Ren TI (2015) Meta-des: a dynamic ensemble selection framework using meta-learning. Pattern Recogn 48(5):1925–1935
11. Dudek G (2016) Pattern-based local linear regression models for short-term load forecasting. Electr Power Syst Res 130:139–147
12. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008) Liblinear: a library for large linear classification. J Mach Learn Res 9(Aug):1871–1874
13. Feng Y, Yang Y, Suykens JA (2016) Robust gradient learning with applications. IEEE Transactions on Neural Networks and Learning Systems 27(4):822–835
14. Friedman J, Hastie T, Tibshirani R et al (2000) Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Ann Stat 28(2):337–407
15. Geman D, Reynolds G (1992) Constrained restoration and the recovery of discontinuities. IEEE Trans Pattern Anal Mach Intell 14(3):367–383
16. Geman D, Yang C (1995) Nonlinear image recovery with half-quadratic regularization. IEEE Trans Image Process 4(7):932–946
17. Genton MG (1998) Highly robust variogram estimation. Math Geol 30(2):213–221
18. Hajiabadi H, Molla-Aliod D, Monsefi R (2017) On extending neural networks with loss ensembles for text classification. In: Proceedings of the Australasian language technology association workshop 2017
19. He R, Zheng WS, Tan T, Sun Z (2014) Half-quadratic-based iterative minimization for robust sparse representation. IEEE Trans Pattern Anal Mach Intell 36(2):261–275
20. Holland MJ, Ikeda K (2016) Minimum proper loss estimators for parametric models. IEEE Trans Signal Process 64(3):704–713
21. Huber PJ et al (1964) Robust estimation of a location parameter. The annals of mathematical statistics 35(1):73–101
22. Islam M, Rojas E, Bergey D, Johnson A, Yodh A (2003) High weight fraction surfactant solubilization of single-wall carbon nanotubes in water. Nano letters 3(2):269–273
23. Kang S, Kang P (2018) Locally linear ensemble for regression. Inf Sci 432:199–209
24. Khan I, Roth PM, Bais A, Bischof H (2013) Semi-supervised image classification with huberized laplacian support vector machines. In: 2013 IEEE 9th international conference on Emerging technologies (ICET), IEEE, pp 1–6.
25. Ko AH, Sabourin R, Britto AS Jr (2008) From dynamic classifier selection to dynamic ensemble selection. Pattern Recogn 41(5):1718–1731
26. Liu W, Pokharel PP, Principe JC (2006) Correntropy: a localized similarity measure. In: 2006. IJCNN'06. International joint conference on Neural networks, IEEE, pp 4919–4924
27. Liu W, Pokharel PP, Príncipe JC (2007) Correntropy: properties and applications in non-gaussian signal processing. IEEE Trans Signal Process 55(11):5286–5298
28. Liu Y, Yao X, Higuchi T (2000) Evolutionary ensembles with negative correlation learning. IEEE Trans Evol Comput 4(4):380–387
29. López J, Maldonado S (2018) Robust twin support vector regression via second-order cone programming. Knowl-Based Syst 152:83–93
30. Mannor S, Meir R (2001) Weak learners and improved rates of convergence in boosting. In: Advances in neural information processing systems, pp 280–286
31. Masnadi-Shirazi H, Mahadevan V, Vasconcelos N (2010) On the design of robust classifiers for computer vision. In: 2010 IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 779–786
32. Masnadi-Shirazi H, Vasconcelos N (2009) On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In: Advances in neural information processing systems, pp 1049–1056
33. Mendes-Moreira J, Soares C, Jorge AM, Sousa JFD (2012) Ensemble approaches for regression: a survey. ACM Comput Surv (CSUR) 45(1):10
34. Meyer CD (2000) Matrix analysis and applied linear algebra, vol 71 Siam
35. Miao Q, Cao Y, Xia G, Gong M, Liu J, Song J (2016) Rboost: label noise-robust boosting algorithm based on a nonconvex loss function and the numerically stable base learners. IEEE Transactions on Neural Networks and Learning Systems 27(11):2216–2228
36. Nápoles G, Falcon R, Papageorgiou E, Bello R, Vanhoof K (2017) Rough cognitive ensembles. Int J Approx Reason 85:79–96
37. Painsky A, Rosset S (2016) Isotonic modeling with non-differentiable loss functions with application to lasso regularization. IEEE Trans Pattern Anal Mach Intell 38(2):308–321
38. Peng J, Guo L, Hu Y, Rao K, Xie Q (2017) Maximum correntropy criterion based regression for multivariate calibration. Chemometr Intell Lab Syst 161:27–33
39. Rodriguez-Galiano V, Sanchez-Castillo M, Chica-Olmo M, Chica-Rivas M (2015) Machine learning predictive models for mineral prospectivity: an evaluation of neural networks, random forest, regression trees and support vector machines. Ore Geol Rev 71:804–818
40. Sangari A, Sethares W (2016) Convergence analysis of two loss functions in soft-max regression. IEEE Trans Signal Process 64(5):1280–1288

41. Schaal S, Atkeson CG, Vijayakumar S (2002) Scalable techniques from nonparametric statistics for real time robot learning. Appl Intell 17(1):49–60
42. Steinwart I, Christmann A (2008) Support vector machines. Springer Science & Business Media
43. Tang L, Tian Y, Yang C, Pardalos PM (2018) Ramp-loss nonparallel support vector regression: robust, sparse and scalable approximation Knowledge-Based Systems
44. Uhlich S, Yang B (2012) Bayesian estimation for nonstandard loss functions using a parametric family of estimators. IEEE Trans Signal Process 60(3):1022–1031
45. Vapnik V (1998) Statistical learning theory, vol 1998. Wiley, New York
46. Vapnik V (2013) The nature of statistical learning theory. Springer science & business media
47. Wang K, Zhong P (2014) Robust non-convex least squares loss function for regression with outliers. Knowl-Based Syst 71:290–302
48. Wang Z, Simoncelli EP, Bovik AC (2003) Multiscale structural similarity for image quality assessment. In: 2004 Conference record of the thirty-seventh asilomar conference on Signals, systems and computers, vol 2. IEEE, pp 1398–1402
49. Xiao Y, Wang H, Xu W (2017) Ramp loss based robust one-class svm. Pattern Recogn Lett 85:15–20
50. Xie L, Yin M, Wang L, Tan F, Yin G (2018) Matrix regression preserving projections for robust feature extraction. Knowledge-Based Systems
51. Zhang J, Chung C, Han Y (2016) Online damping ratio prediction using locally weighted linear regression. IEEE Trans Power Syst 31(3):1954–1962
52. Zhang P, Zhuo T, Zhang Y, Huang H, Chen K (2016) Bayesian tracking fusion framework with online classifier ensemble for immersive visual applications. Multimedia Tools and Applications 75(9):5075–5092
53. Zhang T (2004) Statistical behavior and consistency of classification methods based on convex risk minimization. Ann Stat 32(1):56–85. http://www.jstor.org/stable/3448494
54. Zhang T, Oles FJ (2001) Text categorization based on regularized linear classification methods. Inf Retr 4(1):5–31
55. Zhao L, Mammadov M, Yearwood J (2010) From convex to nonconvex: a loss function analysis for binary classification. In: 2010 IEEE international conference on Data mining workshops (ICDMW), IEEE, pp 1281–1288
56. Zhao S, Chen B, Principe JC (2012) An adaptive kernel width update for correntropy. In: The 2012 international joint conference on Neural networks (IJCNN), IEEE, pp 1–5


**Hamideh Hajiabadi**



**Reza Monsefi**

**Hadi Sadoghi Yazdi**