Communications for Statistical Applications and Methods
2016, Vol. 23, No. 4, 355–362

http://dx.doi.org/10.5351/CSAM.2016.23.4.355
Print ISSN 2287-7843 / Online ISSN 2383-4757

Ensemble approach for improving prediction in kernel regression and classification

Sunwoo Han[a], Seongyun Hwang[b], Seokho Lee[1,b]

[a]Department of Applied Statistics, Yonsei University, Korea;
[b]Department of Statistics, Hankuk University of Foreign Studies, Korea

Abstract

Ensemble methods often help increase prediction ability in various predictive models by combining multiple weak learners and reducing the variability of the final predictive model. In this work, we demonstrate that ensemble methods also enhance the accuracy of prediction under kernel ridge regression and kernel logistic regression classification. Here we apply bagging and random forests to two kernel-based predictive models; and present the procedure of how bagging and random forests can be embedded in kernel-based predictive models. Our proposals are tested under numerous synthetic and real datasets; subsequently, they are compared with plain kernel-based predictive models and their subsampling approach. Numerical studies demonstrate that ensemble approach outperforms plain kernel-based predictive models.

Keywords: bagging, bootstrap, ensemble method, kernel trick, logistic regression, random forest, regression

1. Introduction

The kernel method has been a key technique to effectively deal with a nonlinear relationship between predictors and a response variable in a supervised learning framework. Nonlinearity in regression and classification is quite common in practice; however, it is almost impossible to grasp a functional relation between predictors and response prior to analysis, especially when we deal with multivariate predictors. In such a case, kernel trick becomes a valuable tool because users do not need to specify the explicit form of nonlinearity before modeling and analysis. An enlarged feature set through various nonlinear transformations of predictors might be infinite-dimensional; however, the corresponding optimization can also be solved in a finite-dimensional fashion under kernel trick framework. This kernel trick has become popular with a rise of support vector machines in machine learning discipline; however, kernel trick is a quite general method that can be applied to various statistical modeling in order to address nonlinearity.

We suggest an ensemble approach in kernel-based regression and kernel-based logistic regression classification to enhance prediction performance. Ensemble methods, bagging and random forests often yield a better prediction via combining multiple simple predictive models (in regression) or classifiers (in classification). The idea of the ensemble approach is to combine simple weak submodels that will reduce the variance of the fitted model by weakening the dependency on the sampling variation of training data. This leads to reducing the prediction errors on test data. In this paper,

[1] Corresponding author: Department of Statistics, Hankuk University of Foreign Studies, 81 Oedae-ro, Cheoin-gu, Yongin 17035, Korea. E-mail: lees@hufs.ac.kr

Published 31 July 2016 / journal homepage: http://csam.or.kr

we show how ensemble methods can be embedded in the kernel ridge regression (KRR) and kernel logistic regression (KLR) to compare the approaches with plain kernel-based predictive models.

This paper is organized as follows. In Section 2, we briefly review KRR and KLR for classification. Then, we describe the procedures of how bagging and random forests are embedded in KRR and KLR. The performance of our approaches are demonstrated through simulation studies and various real-world data analyses in Section 3. This paper concludes with some remarks in Section 4.

2. Methodology

2.1. Brief review of kernel ridge regression and kernel logistic regression

Regression and logistic regression find the relation between multiple predictors $\mathbf{x} \in \mathbb{R}^p$ with the conditional expectation of the response (in regression) and the success probability (in logistic regression), such as

$$E(y|\mathbf{x}) = f(\mathbf{x}) \quad \text{or} \quad \log\left(\frac{\Pr(y=1|\mathbf{x})}{\Pr(y=0|\mathbf{x})}\right) = f(\mathbf{x}).$$

The simplest way to model the function $f(\mathbf{x})$ is to use a linear function of the predictors. To incorporate the nonlinearity, we may use basis functions in $f(\mathbf{x})$, like

$$f(\mathbf{x}) = \sum_{j=1}^{d} \theta_j \phi_j(\mathbf{x}).$$

The transformed set of $\mathbf{x}$ by mapping $\mathbf{x} \mapsto \mathbf{\Phi}(\mathbf{x}) := (\phi_1(\mathbf{x}), \ldots, \phi_d(\mathbf{x}))$ is called feature space. The practical difficulty is that we are unable to know which feature space is needed in advance (what kind of basis should be used and how many basis functions are required). Kernel tricks can be a solution to these practical questions. Instead of specifying feature space in advance, we may select a kernel function $K(\cdot, \cdot)$. Using a kernel, the function $f(\mathbf{x})$ can be expressed as

$$f(\mathbf{x}) = \sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x}_i) d_i.$$

It is well known that a kernel function, which requires to meet some properties to be a proper kernel, has its own feature set. So, choosing a particular kernel is equivalent to selecting the corresponding feature set implicitly. We simply compute the kernel and put it into optimization without constructing bases from original predictors directly if we use a kernel function with a flexible basis set. Another good aspect with the kernel method is that optimization becomes a finite dimensional problem, while the corresponding feature set can be infinite-dimensional ($d = +\infty$). Thus, using kernel trick, the optimization becomes the penalized least squares

$$\min_{\mathbf{d}} \sum_{i=1}^{n} \left(y_i - \mathbf{K}_i^T \mathbf{d}\right)^2 + \lambda \mathbf{d}^T \mathbf{K} \mathbf{d} \tag{2.1}$$

for regression, and the penalized minimum negative Bernoulli likelihood

$$\min_{\mathbf{d}} \sum_{i=1}^{n} \ell\left((2y_i - 1)\mathbf{K}_i^T \mathbf{d}\right) + \lambda \mathbf{d}^T \mathbf{K} \mathbf{d} \tag{2.2}$$

for logistic regression. Here, $\mathbf{d} = (d_1, \ldots, d_n)^T$, $\mathbf{K}_i = (K(\mathbf{x}_1, \mathbf{x}_i), \ldots, K(\mathbf{x}_n, \mathbf{x}_i))^T$, $\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1,\ldots,n}$, and $\ell(u) = \log(1 + \exp(-u))$. The former is called KRR and the latter KLR, respectively. The regularization parameter λ is chosen outside the fitting procedure, such as cross validation.

For a given λ, the solution of (2.1) has a closed-form solution $\hat{\mathbf{d}}_\lambda = (\mathbf{K} + \lambda \mathbf{I}_n)^{-1}\mathbf{y}$ with $\mathbf{y} = (y_1, \ldots, y_n)^T$. The solution of (2.2) can be obtained through a standard iterative procedure using quadratic approximation (iteratively reweighted least squares), where the updating formula for $\mathbf{d}$ is given as a solution of ridge-type weighted least squares. This can be easily implemented using any standard statistical software and we used a statistical software R for our implementation.

After learning them from training data, prediction is given using $\hat{f}_\lambda(\mathbf{x}_*) = \mathbf{K}_*^T \hat{\mathbf{d}}_\lambda$ where $\mathbf{x}_*$ is the predictor for a new observation and $\mathbf{K}_* = (K(\mathbf{x}_1, \mathbf{x}_*), \ldots, K(\mathbf{x}_n, \mathbf{x}_*))^T$. For regression the predicted value of the response for $\mathbf{x}_*$ is $\hat{y}_* = \hat{f}_\lambda(\mathbf{x}_*)$, and for logistic regression the predicted class label for $\mathbf{x}_*$ is $\{\text{sign}(\hat{f}_\lambda(\mathbf{x}_*)) + 1\}/2$. The details of concept and implementation on kernel-trick can be found in Schölkopf and Smola (2002).

2.2. Brief review of ensemble approaches

Ensemble methods have become a major learning paradigm since the 1990s. This learning approach comprises many methodologies that include bagging, random forests, boosting and their variants. They have been extensively applied in vast application areas that include bioinformatics, cognitive science, natural language processing, and computer vision. An extensive research on ensemble learning and its numerous applications can be found in Bühlmann (2012), Zhou (2012), and references therein.

We consider two ensemble approaches of bagging and random forest for regression and logistic regression classification. Both methods have been introduced in tree-based regression and classification to increase predictability by combining sub-trees from bootstrap samples. However, they are generative approaches that can be generalized to a wide range of predictive modeling. We describe their idea here as a generic approach.

Suppose there is a training data $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, n\}$. From this training data, we draw bootstrap samples, B times, of the same size of $\mathcal{D}$ by allowing replacement, say $\mathcal{D}_1, \ldots, \mathcal{D}_B$. Instead of a single fit from $\mathcal{D}$, bagging uses B fits, $\hat{f}_b(\mathbf{x})$ $(b = 1, \ldots, B)$, each of which is a fit from the b^{th} bootstrap sample $\mathcal{D}_b$, for prediction. In regression, bagging averages $\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_B(\mathbf{x})$ by $\hat{f}_{\text{bagging}}(\mathbf{x}) = \sum_{b=1}^B \hat{f}_b(\mathbf{x})/B$ and use it for a predicted value of y at $\mathbf{x}$. For classification, a classifier of bagging is defined by $\{\text{sign}(\sum_{b=1}^B \text{sign}(\hat{f}_b(\mathbf{x}))/B) + 1\}/2$ if we follow a majority vote scheme. An idea behind bagging is that B fits of $\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_B(\mathbf{x})$ from bootstrap samples $\mathcal{D}_b$ from the same training data $\mathcal{D}$ are roughly identically distributed. They are unbiased if $\mathcal{D}$ is a random sample from a population; however, they use less information and have a predictive power weaker than a single fit from $\mathcal{D}$. By averaging multiple weak predictive models or classifiers, a combined one is still unbiased but can markedly reduce its variability. This leads to enhancing predictability for new observations.

Random forests also use multiple weak fits from the bootstrap samples and define a predictive model (or a classifier) the same as bagging. Different from bagging, bootstrap samples in random forests include a small random subset among p predictors. Therefore, B bootstrap samples $\mathcal{D}_b$ $(b = 1, \ldots, B)$ comprise $m\,(< p)$ predictors which are different across $b = 1, \ldots, B$. The corresponding fits $\hat{f}_b(\mathbf{x})$ are less likely correlated with each other since $\mathcal{D}_1, \ldots, \mathcal{D}_B$ have different set of predictors. This decorrelation greatly reduces the variability of the average of B weak predictive models and classifiers, leading to prediction improvement.

Although bagging and random forests are developed for tree model, they are straightforwardly generalizable to many general predictive modelings. In the following subsection we propose kernel

regression and kernel logistic regression classification using ensemble embedment. We refer to Hastie *et al.* (2011) for an introduction to ensemble approaches in learning.

2.3. Ensemble approach in kernel regression and kernel logistic regression

In this paper, we embrace two ensemble methods of bagging and random forests in KRR and KLR classification. This embracement is quite straightforward and we describe how we implement them in the following simulation and data analysis.

First, we randomly draw a bootstrap sample from the training data with replacement, resulting in the same sample size of the bootstrap sample equal to the sample size of the training data. We denote the bootstrap sample as $\mathcal{D}_b = \{(\mathbf{x}_i^b, y_i^b) \mid i = 1, 2, \ldots, n\}$. Bootstrapping is repeated B times and let denote them $\mathcal{D}_1, \ldots, \mathcal{D}_B$. In the following simulation and data analysis we use $B = 500$. For the b^{th} bootstrap sample ($b = 1, \ldots, B$), we compute the kernel $K(\mathbf{x}_i^b, \mathbf{x}_{i'}^b)$ for all pairs of (i, i') with $i, i' = 1, \ldots, n$. Then, for a given value of λ, we obtain $\hat{\mathbf{d}}_\lambda^b$ through the minimization of (2.1) for KRR or (2.2) for KLR classification. For a new data $\mathbf{x}_*$, we compute the kernel function $\mathbf{K}_{b,\text{bagging}} = (K(\mathbf{x}_1^b, \mathbf{x}_*), \ldots, K(\mathbf{x}_n^b, \mathbf{x}_*))^T$ between $\mathbf{x}^b$ in the b^{th} bootstrap sample and the new data and, then, calculate the function $\hat{f}_{b,\text{bagging}}(\mathbf{x}_*) = \mathbf{K}_{b,\text{bagging}}^T \hat{\mathbf{d}}_\lambda^b$ for $b = 1, \ldots, B$.

For KRR, bagging prediction at $\mathbf{x}_*$ is given by $\hat{y}_{\text{bagging}} = \sum_{b=1}^B \hat{f}_{b,\text{bagging}}(\mathbf{x}_*)/B$. However, we can use three different ways for class prediction for kernel logistic ridge regression. The first is to use the average of logit estimates for prediction, where we compute $\hat{f}_{\text{bagging}}(\mathbf{x}_*) = \sum_{b=1}^B \hat{f}_{b,\text{bagging}}(\mathbf{x}_*)/B$ and use it for class prediction as $\hat{y}_{\text{bagging}} = \{\text{sign}(\hat{f}_{\text{bagging}}(\mathbf{x}_*)) + 1\}/2$. The second is to use the average of probability estimates, where we first calculate the probability of $y = 1$ for $b = 1, \ldots, B$ as $\hat{p}_b = 1/\{1 + \exp(-\hat{f}_{b,\text{bagging}}(\mathbf{x}_*))\}$ and then the class label is predicted as $\hat{y}_{\text{bagging}} = \{\text{sign}((1/B)\sum_{b=1}^B \hat{p}_b - 1/2) + 1\}/2$. The last one is to use the majority vote scheme where the class prediction is given by $\{\text{sign}(\sum_{b=1}^B \text{sign}(\hat{f}_{b,\text{bagging}}(\mathbf{x}_*))/B) + 1\}/2$. All three ways of prediction in the above are possible for classification, but in the following section, we only provide the results of the first one, average of logit estimates scheme, to keep brevity while other two schemes show the similar performance in classification (Hwang, 2016).

Prediction under random forests is similar to bagging prediction. For the b^{th} bootstrap sample, we randomly select m predictors from $\mathbf{x}_i^b$ having p predictors, which is denoted by $\mathbf{z}_i^b \in \mathbb{R}^m$. In our implementation we used $m \approx \sqrt{p}$. Thus, in computing kernel using bootstrap sample, we use $\mathbf{z}^b$ to calculate $K(\mathbf{z}_i^b, \mathbf{z}_{i'}^b)$ for $i, i' = 1, \ldots, n$ across the bootstrap sample and $\mathbf{K}_{b,\text{rf}} = (K(\mathbf{z}_1^b, \mathbf{z}_*), \ldots, K(\mathbf{z}_n^b, \mathbf{z}_*))^T$ between the bootstrap samples and a new observation $\mathbf{x}_*$, where $\mathbf{z}_*$ is the collection of m predictors from $\mathbf{x}_*$. Note that the set of m selected predictors is different across bootstrap samples. For prediction we compute the function $\hat{f}_{b,\text{rf}}(\mathbf{x}_*) = \mathbf{K}_{b,\text{rf}}^T \hat{\mathbf{d}}_\lambda^b$ for $b = 1, \ldots, B$. The remaining procedure for prediction $\hat{y}_{\text{rf}}$ in regression and classification is the same as the bagging approach described above.

Various types of kernel function can be applied to kernel-based regression and classification; however, in this implementation, we use a radial basis kernel whose form is $K(\mathbf{x}, \mathbf{x}') = \exp(-\sigma\|\mathbf{x} - \mathbf{x}'\|^2)$. A parameter σ inside the kernel can be optimally chosen based on prediction performance; however, we set $\sigma = 1/p$ throughout implementation. This choice is for the consideration of computational costs that are also widely used in practice. However, it does not affect the performance comparison in the following section.

The ridge penalty parameter λ in (2.1) and (2.2) can be chosen when a validation data, which is independent of training data, is available. Or typical K-fold cross-validation is also possible for choosing λ. However, one of the good features of bagging and random forests is that each bootstrap sampling procedure omits the unselected observations remaining in the training data (*out-of-bag ob-*

servations) that are not used in the fitting procedure. Out-of-bag (OOB) observations can be used as a validation data for the selection of λ (Breiman, 1996). The optimal λ is chosen as a minimizer of OOB mean squared errors for the KRR and OOB misclassification error rate for KLR classification, respectively. OOB error calculation is cheaper than cross-validation error calculation.

We summarize the whole procedure into the below conceptual algorithm:

Algorithm 1: Ensemble method in kernel-based regression and classification

1. Set $\sigma = 1/p$ (bagging) or $\sigma = 1/m$ (random forest)
2. For $b = 1, \ldots, B$ and for λ on a pre-specified grid, repeat
   (a) Create a bootstrap sample $\mathcal{D}_b$, and set an OOB sample O_b that consists of the observations not included in $\mathcal{D}_b$. In this step, $\mathcal{D}_b$ and O_b contains randomly selected $m \approx \sqrt{p}$ predictors for random forest, $m = p$ predictors for bagging.
   (b) Compute the kernel matrix $\mathbf{K}_b$ on $\mathcal{D}_b$, and calculate $\hat{\mathbf{d}}_\lambda^b$.
   (c) Compute the kernel matrix $\mathbf{K}_b^*$ between $\mathcal{D}_b$ and O_b, and compute $\hat{f}_{b,\lambda}(\mathbf{x}_i)$ for $\mathbf{x}_i \in O_b$.
   (d) Compute an OOB error, say $\mathrm{OE}_b(\lambda)$, by applying $\hat{f}_{b,\lambda}(\mathbf{x}_i)$ for $\mathbf{x}_i \in O_b$.
3. Compute $\mathrm{OE}(\lambda) = (1/B)\sum_{b=1}^B \mathrm{OE}_b(\lambda)$, and find the optimal λ as $\lambda_{opt} = \arg\min_\lambda \mathrm{OE}(\lambda)$.
4. Given a new observation $\mathbf{x}_*$, for $b = 1, \ldots, B$, repeat
   (a) Obtain $\hat{\mathbf{d}}_{\lambda_{opt}}^b$ using $\mathbf{K}_b$ and $\mathcal{D}_b$ used in Step 2.
   (b) Compute $\mathbf{K}_b(\mathbf{x}_*) = (K(\mathbf{x}_1, \mathbf{x}_*), \ldots, K(\mathbf{x}_n, \mathbf{x}_*))^T$ and compute $\hat{f}_{b,\lambda_{opt}}(\mathbf{x}_*)$.
   (c) Compute $\hat{f}_{\lambda_{opt}}(\mathbf{x}_*) = (1/B)\sum_{b=1}^B \hat{f}_{b,\lambda_{opt}}(\mathbf{x}_*)$
5. Report $\hat{y} = \hat{f}_{\lambda_{opt}}(\mathbf{x}_*)$ for regression and $\hat{y} = \{\mathrm{sign}(\hat{f}_{\lambda_{opt}}(\mathbf{x}_*)) + 1\}/2$ for classification as the prediction for $\mathbf{x}_*$.

3. Numerical studies

We tested our proposals using simulated data and real data. For the comparison purpose, we conducted a vanilla version of KRR and KLR throughout the numerical studies. Since there is no OOB observations in KRR and KLR, λ is selected through 5-fold cross-validation. We additionally conducted a subsampling approach for kernel ridge regression (KRR-sub) and kernel logistic regression (KLR-sub), where the predictive model and classifier is developed under the best subsample of the training data that yields the minimum OOB prediction error (Huh, 2015). The subsample size in this approach is roughly 70% of the training data and the best subsample is searched in over 500 randomly selected subsamples. In the following, we denote our approaches for kernel ridge regression and kernel logistic regression by (KRR-bagging, KRR-rf) and (KLR-bagging, KLR-rf), respectively, where appendages that follow the hyphen represent the type of ensemble methods used.

Table 1: Average of 100 test RMSEs and standard error (in parenthesis) for regression

p	n	Method			
		KRR	KRR-sub	KRR-bagging	KRR-rf
3	50	0.7037 (0.0060)	0.7203 (0.0062)	0.6928 (0.0049)	0.6602 (0.0044)
	100	0.6250 (0.0036)	0.6415 (0.0042)	0.6188 (0.0036)	0.6157 (0.0032)
	200	0.5695 (0.0019)	0.5811 (0.0021)	0.5669 (0.0020)	0.5903 (0.0021)
	400	0.5427 (0.0014)	0.5512 (0.0016)	0.5412 (0.0015)	0.5762 (0.0018)
5	50	0.9016 (0.0065)	0.9329 (0.0086)	0.9060 (0.0059)	0.8045 (0.0050)
	100	0.7787 (0.0049)	0.8145 (0.0058)	0.7861 (0.0044)	0.7259 (0.0037)
	200	0.6921 (0.0028)	0.7254 (0.0038)	0.6902 (0.0025)	0.6742 (0.0025)
	400	0.6329 (0.0020)	0.6566 (0.0023)	0.6272 (0.0019)	0.6426 (0.0020)
10	50	1.2211 (0.0067)	1.2570 (0.0086)	1.2491 (0.0062)	1.0512 (0.0052)
	100	1.1069 (0.0058)	1.1448 (0.0055)	1.1295 (0.0051)	0.9549 (0.0043)
	200	0.9839 (0.0035)	1.0200 (0.0041)	1.0000 (0.0035)	0.8876 (0.0032)
	400	0.8784 (0.0029)	0.9075 (0.0037)	0.8863 (0.0030)	0.8418 (0.0023)
20	50	1.4177 (0.0059)	1.4518 (0.0076)	1.4681 (0.0064)	1.2898 (0.0056)
	100	1.3381 (0.0045)	1.3619 (0.0052)	1.3600 (0.0047)	1.1905 (0.0039)
	200	1.2679 (0.0036)	1.2925 (0.0037)	1.2785 (0.0034)	1.1255 (0.0029)
	400	1.1723 (0.0033)	1.2101 (0.0037)	1.1918 (0.0032)	1.0826 (0.0028)
40	50	1.5626 (0.0064)	1.5916 (0.0087)	1.6324 (0.0071)	1.5073 (0.0066)
	100	1.4744 (0.0047)	1.5010 (0.0055)	1.5082 (0.0050)	1.4036 (0.0046)
	200	1.4143 (0.0046)	1.4360 (0.0047)	1.4284 (0.0043)	1.3312 (0.0033)
	400	1.3607 (0.0037)	1.3780 (0.0043)	1.3673 (0.0038)	1.2907 (0.0034)

The best performer for each case is highlighted. KRR = kernel ridge regression.

3.1. Simulation

We generated x_{ij} ($i = 1, \ldots, n$ and $j = 1, \ldots, p$) randomly from uniform distribution over $(-2, 2)$. The continuous-type response variable was drawn as $y_i \sim N(\sum_{j=1}^{p}(x_{ij}/2)^j, (0.5)^2)$. We consider the number of predictors, p, as 3, 5, 10, 20, 40. Sample size of the training data, n, is considered 50, 100, 200, 400. Test dataset was generated following the same data-generating scheme, but its size is fixed as 1,000. We conducted the vanilla, subsampling, bagging, random forests versions of KRR to the training data, and applied the predictive models from those methods to the test data in order to compute root mean squared errors (RMSE) for test data. This procedure was repeated 100 times to provide the average and its standard error (Table 1). Table 1 indicates that bagging and random forests approaches show smaller RMSEs than vanilla or subsampling versions in all cases. This result demonstrates that ensemble approaches improve the predictive performance in KRR.

For classification, we generated x_{ij} randomly from $N(0, 10^2)$ and we computed the success probability $p_i = \exp\{-1/(\lfloor p/2 \rfloor \times 10^2) \sum_{j=1}^{\lfloor p/2 \rfloor} x_{ij}^2\}$ for $i = 1, \ldots, n$. Using this success probability, binary class label y_i was randomly generated from Bernoulli distribution of p_i. Predicting class label becomes a hard problem since only half of p predictors are used to construct success probability in this simulation. The decision boundary from this data-generating scheme is also of a circular shape and highly nonlinear. We set the number of predictors, training sample sizes, and test sample sizes the same as in the previous simulation for regression. Table 2 provides the average of test misclassification error rates. Predicting class label for test data follows the averaging logit estimates scheme in Table 2; however, two other schemes can also be possible and their results can be found in Hwang (2016). Table 2 indicates that the ensemble methods also outperform others in the classification. Note that KLR and KLR-sub show that their misclassification error rates are larger than a half for $p = 10, 20, 40$ cases and imply that classification fails totally. However, ensemble methods improve performance even though the classification of this simulation is a tough task.

Table 2: Average of 100 test misclassification rates and its standard error (in parenthesis) for classification

p	n	Method			
		KLR	KLR-sub	KLR-bagging	KLR-rf
3	50	0.4207 (0.0034)	0.4430 (0.0037)	0.3816 (0.0028)	0.4101 (0.0022)
	100	0.3754 (0.0023)	0.3975 (0.0026)	0.3529 (0.0022)	0.3886 (0.0022)
	200	0.3519 (0.0020)	0.3632 (0.0023)	0.3402 (0.0020)	0.3678 (0.0020)
	400	0.3305 (0.0018)	0.3408 (0.0019)	0.3246 (0.0018)	0.3369 (0.0018)
5	50	0.4556 (0.0025)	0.4627 (0.0026)	0.4296 (0.0024)	0.4312 (0.0022)
	100	0.4328 (0.0022)	0.4441 (0.0022)	0.4053 (0.0022)	0.4119 (0.0020)
	200	0.4172 (0.0018)	0.4259 (0.0023)	0.3943 (0.0017)	0.3946 (0.0017)
	400	0.3983 (0.0020)	0.4076 (0.0020)	0.3828 (0.0020)	0.3776 (0.0017)
10	50	0.5707 (0.0016)	0.5704 (0.0016)	0.4963 (0.0018)	0.4759 (0.0019)
	100	0.5714 (0.0016)	0.5716 (0.0016)	0.4926 (0.0016)	0.4718 (0.0018)
	200	0.5705 (0.0016)	0.5735 (0.0017)	0.4881 (0.0017)	0.4632 (0.0020)
	400	0.5604 (0.0018)	0.5660 (0.0017)	0.4800 (0.0016)	0.4533 (0.0017)
20	50	0.5928 (0.0016)	0.5928 (0.0016)	0.5013 (0.0015)	0.4755 (0.0028)
	100	0.5929 (0.0016)	0.5929 (0.0016)	0.4997 (0.0016)	0.4711 (0.0021)
	200	0.5928 (0.0016)	0.5928 (0.0016)	0.5040 (0.0016)	0.4656 (0.0022)
	400	0.5931 (0.0016)	0.5932 (0.0016)	0.4982 (0.0018)	0.4561 (0.0018)
40	50	0.6130 (0.0016)	0.6130 (0.0016)	0.4999 (0.0015)	0.4771 (0.0024)
	100	0.6130 (0.0016)	0.6130 (0.0016)	0.5014 (0.0015)	0.4735 (0.0022)
	200	0.6130 (0.0016)	0.6130 (0.0016)	0.4995 (0.0017)	0.4732 (0.0021)
	400	0.6130 (0.0016)	0.6130 (0.0016)	0.4995 (0.0016)	0.4688 (0.0020)

The best performer for each case is highlighted. KLR = kernel logistic regression.

Table 3: Real data analysis results

Type of analysis	Dataset	p	n	Method			
				KRR	KRR-sub	KRR-bagging	KRR-rf
Regression	Airfoil self-noise	5	1503	47.26 (0.22)	56.62 (0.19)	55.62 (0.18)	11.17 (0.10)
	Concrete slump test	8	103	35.21 (0.06)	36.13 (0.06)	35.83 (0.06)	21.65 (0.08)
	Energy efficiency	8	758	1.55 (0.03)	1.73 (0.03)	1.52 (0.02)	2.37 (0.02)
	Computer hardware	6	209	175.60 (2.84)	176.09 (2.83)	176.67 (2.84)	148.36 (3.27)
	Auto MPG	7	398	24.42 (0.04)	24.47 (0.04)	24.50 (0.04)	20.04 (1.83)
	Yacht hydrodynamics	6	308	5.12 (0.06)	5.47 (0.06)	5.08 (0.06)	8.77 (0.07)
	Housing	13	506	22.71 (0.05)	23.04 (0.06)	23.13 (0.05)	11.06 (0.07)
Classification	Cylinder bands	23	358	0.401 (0.004)	0.444 (0.003)	0.387 (0.003)	0.328 (0.005)
	Forest type mapping	27	326	0.199 (0.003)	0.219 (0.003)	0.160 (0.002)	0.159 (0.003)
	Dow Jones index	13	720	0.542 (0.002)	0.542 (0.002)	0.500 (0.002)	0.407 (0.002)
	Haberman's survival	3	306	0.314 (0.003)	0.320 (0.004)	0.296 (0.003)	0.275 (0.003)
	Ionosphere	34	351	0.119 (0.003)	0.120 (0.003)	0.127 (0.003)	0.068 (0.002)
	Pima Indians diabetes	8	768	0.389 (0.002)	0.407 (0.002)	0.377 (0.002)	0.327 (0.002)
	Heart	5	270	0.353 (0.003)	0.349 (0.003)	0.430 (0.004)	0.342 (0.003)
	Blood transfusion	4	748	0.302 (0.002)	0.313 (0.002)	0.291 (0.002)	0.239 (0.002)
	Breast tissue	9	106	0.478 (0.004)	0.479 (0.005)	0.495 (0.007)	0.354 (0.007)
	Urban land cover	147	168	0.545 (0.004)	0.545 (0.004)	0.493 (0.006)	0.250 (0.005)
	Australian credit approval	5	502	0.354 (0.002)	0.351 (0.002)	0.453 (0.003)	0.298 (0.002)

Average of 100 test rooted mean squared errors or misclassification rates and its standard error (in parenthesis) are provided for regression and classification, respectively. The best performer for each case is highlighted. KRR = kernel ridge regression; KLR = kernel logistic regression.

3.2. Real data application

We applied all methods used in the simulation study to several real-world datasets from the UCI machine learning repository (Lichman, 2013) listed in Table 3. Before analysis, we removed some samples with missing values and some nominal categorical predictors from the datasets. Seven datasets

are used for regression and 11 datasets for classification respectively. For each dataset, we evenly split the dataset into two parts, training dataset and test dataset, in random. Four methods were applied to the training data to develop a predictive model or classifier; subsequently, the test dataset was then used to evaluate test RMSE or misclassification rate. To alleviate the dependence on random split, we conducted this procedure 100 times and report average of 100 RMSEs and misclassification rates in the Table 3. The results in Table 3 demonstrates that ensemble methods are quite competitive in kernel-based learning problems.

4. Conclusion and remarks

In this paper, we propose to apply ensemble methods in kernel-based regression and classification. Throughout some limited simulation studies and real data analyses, we demonstrate their competitiveness numerically. We conclude this paper with some remarks. First, we confine the scope of methods on bagging and random forests in this study. However, boosting approach can also be applied naturally to our framework. Boosting develops a final predictive model by adding incrementally numerous submodels that typically have very weak predictive power. Our framework can naturally and easily embrace a boosting technique into kernel-based learning and we leave it to future research. Second, there is a concern that bagging and random forests are computationally burdensome. This practical problem can be resolved through the current development on computing environment. Their implementation is naturally suitable for parallel computing since subfittings on bootstrap samples under bagging and random forests scheme are exclusively independent with each other.

Acknowledgments

This research was supported by Hankuk University of Foreign Studies Research Fund of 2015.

References

Breiman L (1996). *Out-of-bag estimation*, Technical report, Department of Statistics, University of California at Berkeley, CA, USA.

Bühlmann P (2012). Bagging, boosting and ensemble methods. In Gentle JE, Härdle WK, Mori Y (Eds), *Handbook of Computational Statistics* (pp. 985–1022). Springer, Heidelberg.

Han S (2016). *A study on efficiency of kernel ridge regression using ensemble methods* (Master's thesis), Hankuk University of Foreign Studies, Yongin, Korea.

Hastie T, Tibshirani R, and Friedman J (2011). *The Elements of Statistical Learning* (2nd ed.), Springer, New York.

Huh MH (2015). Kernel-trick regression and classification, *Communications for Statistical Applications and Methods*, **22**, 201–207.

Hwang S (2016). *A study on efficiency of kernel ridge logistic regression classification using ensemble method* (Master's thesis), Hankuk University of Foreign Studies, Yongin, Korea.

Lichman M (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Schölkopf B and Smola AJ (2002). *Learning with Kernels*, The MIT Press, Cambridge, MA.

Zhou ZH (2012). *Ensemble Methods: Foundations and Algorithms*, CRC Press, Boca Raton, FL.

Received June 25, 2016; Revised July 7, 2016; Accepted July 8, 2016