Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference

An Ensemble Approach to Instance-Based Regression Using Stretched Neighborhoods

Vahid Jalali and David Leake
School of Informatics and Computing, Indiana University
Informatics West, 901 E. 10th Street
Bloomington, Indiana, 47408

Abstract

Instance-based regression methods generate solutions from prior solutions within a neighborhood of the input query. Their performance depends on both neighborhood selection criteria and on the method for generating new solutions from the values of prior instances. This paper proposes a new approach to addressing both problems, in which solutions are generated by an ensemble of solutions of local linear regression models built for a collection of "stretched" neighborhoods of the query. Each neighborhood is generated by relaxing a different dimension of the problem space. The rationale is to enable major change trends along that dimension to have increased influence on the corresponding model. The approach is evaluated for two candidate relaxation approaches, gradient-based and based on fixed profiles, and compared to baselines of k-NN and using a radius-based spherical neighborhood in n-dimensional space. Results in four test domains show up to 15 percent improvement over baselines, and suggest that the approach could be particularly useful in domains for which the space of prior instances is sparse.

Introduction

Lazy learning methods postpone building a model or making an estimation for the target function until a query is submitted, generating local estimates tailored the specific input problems. Lazy learning can be especially beneficial for complex and incomplete domains in which a set of (possibly relatively simpler) local models may provide higher quality results than a single global model. The notion of locality in lazy learning is often defined based on a distance function that is used for finding a set of nearest neighbors for the input query, from whose values a value is computed for the input query.

Normally the neighborhood is determined by the distance function and a predefined number of nearby instances to consider: *k-NN* considers the k nearest neighbors. However, other neighborhood selection schemes and combination functions may sometimes be more appropriate. In this paper, we explore an approach using new neighborhood selection functions for choosing points from which to calculate a target value, using linear regression to develop local models, and then combining the values of those models to produce a value. From the perspective of case-based reasoning, this corresponds to a new approach to adapting the solutions of prior cases for regression tasks.

The method trains linear regression models for local neighborhoods of the input query, in which the nearest neighbor calculation is adapted to "stretch" one dimension. More specifically, for an n-dimensional space, distances are computed by a normal domain similarity metric in n-1 dimensions, and distances in one dimension are relaxed according to either a gradient-based strategy or a strategy based on a fixed "shape" profile. By generating a model for a set neighborhoods, each relaxing of one of the n possible dimensions, n different linear regression models are generated for the specific query. The final value is generated by averaging the estimate returned by each of the models. We first present the approach and then evaluate its performance for four sample domains, showing encouraging results. We close with observations and topics for future research.

Related Work

Many instance-based learning approaches use *k-NN*, selecting neighborhoods composed of the k nearest instances. The effects of neighborhood shape have received little study.

Outside of instance-based learning, numerous approaches have been explored for regression tasks. For example, Kwok and Yeung (yau Kwok and Yeung 1997) apply feed-forward neural networks, Orr (Orr 1996) applies radial basis function networks, and Scholkopf and Smola (Scholkopf and Smola 2001) apply Support Vector Machines by transforming the regression problem into a constrained optimization problem. These methods differ from our work both in the utilized models and the non-lazy nature of the model generation.

Within case-based reasoning, McSherry (McSherry 1998) proposes a case-based reasoning regression approach based on pair-wise comparisons between cases in a case-base and using those pairs for adapting the solution from a retrieved case for an input query. Most relevant to our approach, Patterson et. al. (Patterson, Rooney, and Galushka 2002) train a locally weighted regression model for predicting the difference in the target value of two cases. They use a distance weighted average for creating a generalized case from the top k nearest neighbors to the input query, and adapt the so-

lution from the generalized case by feeding the differences between the input query and the generalized case into the trained linear regression model and adding the estimated difference to the retrieved solution. However, instead of using linear regression for adapting the solution from a k-NN component, we directly use linear regression for predicting the target value of an input query.

Linear regression is widely used, but in some domains, the relation between the input features and the target function is non-linear, so multiple local linear models can yield more accurate estimations compared to a global linear model. Atkeson (Atkeson, Moore, and Schaal 1997) studies aspects of locally weighted linear regression such as distance functions, smoothing parameters, weighting functions, local model structures and other related issues. However, the distance functions and consequently the neighborhoods used in our methods are different from those discussed in Atkeson's work and we explore using an ensemble of locally weighted linear regression models.

Ensemble methods have been the subject of much study. These methods (e.g. Boosting (Schapire 1990) or Bagging (Breiman 1996)) combine estimations from different models to improve performance. Freunnd and Schapire's (Freund and Schapire 1997) AdaBoost works by generating a set of weak models and assigning greater weights to examples with unsatisfactory predicted values. It uses a weighted majority vote of a set of weak hypotheses for estimating the target value of the input queries. There are many variations of boosting (Drucker 1997; Friedman 2000; Ye et al. 2009) and bagging (Sun and Pfahringer 2011) that usually differ in the loss function they use or the gradient direction they explore. In contrast to the ensemble methods discussed so far where the base models are eager, there are other alternative methods (e.g. (Zhu and Yang 2008) and (Zhu, Bao, and Qiu 2008)) that use locally weighted base models. The proposed methods in this paper are lazy bagging methods. They differ from other similar methods in that they use a different (overlapping) non-spherical-shaped neighborhood for training each base model while other similar methods use a fixed set of instances selected from a spherical-shaped neighborhood and train base models on different sub-sets of those instances.

Linear Regression

Linear Regression (LR) generates a linear target function, based on a set of features to consider, from a collection of input instances. Applying linear regression to local neighborhoods requires choosing the instances for training the model (i.e., selecting the neighborhoods). Applying an ensemble of linear regression models requires selecting a collection of possible models and determining how to combine their values. This section begins with brief background on locally weighted regression, which our approach applies to the cases in a neighborhood to build local models, and then considers the questions of neighborhood selection and exploitation of the set of models.

Locally Weighted Regression

Given a set of instances to constitute the neighborhood of a query, Locally Weighted Regression (LWR) generates a regression model based on weighting points according to their distance from the query, so that the points far away from the input query contribute less in determining the model compared to the nearby points. Let x_i and y_i represent the i^{th} training sample and its value respectively. If $f : X \rightarrow R$ is a function that maps an instance to its estimated value and q represents the input query, then LWR uses the following criterion C for model selection:

$$C \equiv \sum_{i=1}^{n} (f(x_i) - y_i)^2 K(d(x_i, q)) \quad (1)$$

where $d(x_i, q)$ indicates the distance (e.g. Euclidean distance) between the i^{th} training sample and the query and K is the kernel function used for distance weighting. For example, the kernel function could be chosen so that only a few nearby points have significant effect.

Profile-Based Axis Relaxation for Neighborhood Selection

The motivation for our relaxation-based approach is to make instance-based regression more sensitive to trends which may exist outside of a constant-radius neighborhood or which may arise in a direction outside the majority of the instances in a radius-based neighborhood. The goal is to enable interpolation from points which reflect important trends but might not be reflected by traditional neighborhoods.

As a baseline, we consider spherical-based neighborhood selection method. In addition, we consider three methods which relax dimensions along particular shape profiles, which we describe intuitively as cylindrical, hourglass, and diamond-shaped. Examples are illustrated, as they apply to 2-dimensional space in Fig. 1. Within each of these basic shapes a family of neighborhoods can be defined, as follows:

1. Spherical: All points within a fixed radius of the input query are included.
2. Cylindrical: If x_i and q_i represent the value of the i^{th} feature of a stored instance and of the query, respectively, we delete the d^{th} feature and apply Minkowski's metric, as follows:

$$d1(x, q) \equiv \left(\sum_{i \in \{1,..,n\} - \{d\}} (\mid x_i - q_i \mid)^p \right)^{\frac{1}{p}} \quad (2)$$

   Note that if p is set to 2, this calculates Euclidean distance in the n-1 dimensional space resulting from ignoring the d^{th} feature.
3. Hourglass: The level of attention to points along dimension d can be varied by applying Eq. 2 (using 2 as p) as the distance function in Eq. 3 and incorporating a kernel function $K1$ based on the values of x_d and q_d, as follows:

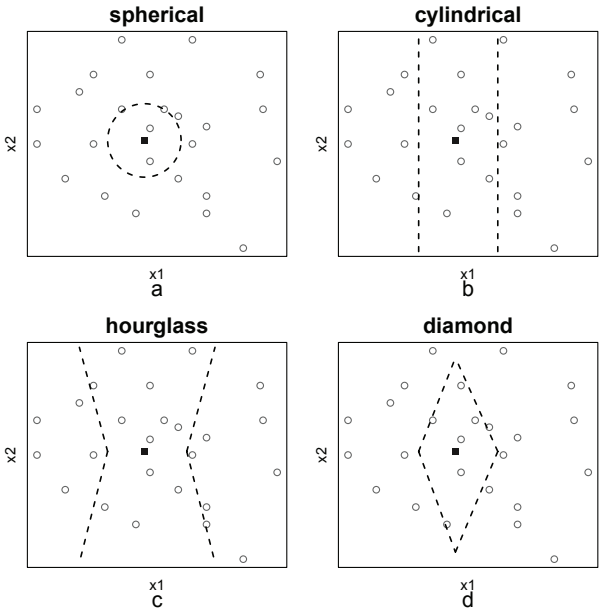

Figure 1: Four different neighborhoods in 2-d space

$$d2(x,q) \equiv \sqrt{\sum_{i \in \{1,..,n\}-\{d\}} (x_i - q_i)^2} \times K1(\mid x_d - q_d \mid) \tag{3}$$

If K1 is inversely related to the difference between x_d and q_d (e.g. $K1 = \frac{1}{|x_d - q_d|} + C$ where C is a constant) then a neighborhood like the one depicted at part c of Fig. 1 is selected.

4. Diamond: If K1 in Eq. 3 is directly related to the difference between x_d and q_d (e.g. $K1 = \mid x_d - q_d \mid + C$ where C is a constant), then a neighborhood such as shown in part d of Fig. 1 is selected. Using the diamond neighborhood decreases the chance of selecting points distant from the input query in the direction of the relaxed dimension, compared to a cylindrical or hourglass neighborhood.

Gradient-based Weighted Axis Relaxation

In addition to the profile-based relaxations illustrated in the previous section, we consider another method which directly uses gradients to attempt to stretch the neighborhood to include points which better represent a trend (either ascending or descending) for the target function over the relaxed dimension. For this we modify the distance function from Eq. 3 so that the trend-representative points contribute more in training the model:

$$d3(x,q) \equiv \sqrt{\sum_{i \in \{1,..,n\}-\{d\}} (x_i - q_i)^2} \times K2(n(x_d), n(y)) \tag{4}$$

Here all variables are as used in Eq. 3, y represents the actual value of the target function at point x and $n()$ is a normalizer function. $K2$ is a kernel function that assigns more weight to those points that can capture a trend over the d^{th} dimension. $K2$ could be selected so that either points on a descending or points on an ascending trend over the d^{th} dimension are weighted more compared to the other datapoints. E.g., favoring data points on an ascending trend can be achieved with the kernel function:

$$K2(x_i, y) \equiv \mid y - x_i \mid + E_i \tag{5}$$

Here x and y are normalized (between 0 and 1) values of the i^{th} attribute of x and its actual target values respectively and E_i is a value used for biasing the effect of $K2$ in Eq. 4. E_i can vary for different points or could be a constant value. Fig. 2 depicts a sample $K2$ weighting function where E_i is always equal to zero. In Fig. 2, the x and y axes represent the normalized attribute and case values respectively as explained in Eq. 5. $K2$ axis is the kernel function used.

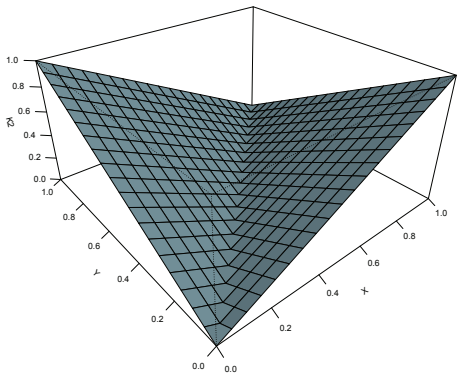

Figure 2: A sample K2 weighting function when E = 0 for all points.

Ensemble Method for Combining Model Predictions

We hypothesize that instance-based regression can be improved by training as many locally weighted regression models as the number of dimensions, with estimations from those models combined to form the final estimation of the target function. If $f_d(q)$ is the estimation from the model trained based on a neighborhood defined using Eq. 3 by stretching the d^{th} dimension, and w is a vector used for weighting the effect of each model in determining the final value of the target value, then the following formula can be used for calculating the final estimation of the target value:

$$f(q) \equiv \sum_{d=1}^{n} f_d(q) w_d \tag{6}$$

Weights could be set based on different criteria such as the quality of the trained models (e.g. in terms of Sum

of Squared Residuals for the training data) or the coefficient for each dimension obtained from a linear regression model weighted locally according to the radius-based spherical neighborhood in n-dimensional space. If all elements of w are equal, then $f(q)$ would simply be the mean of the estimations. In our following experiments, except the number of neighbors to use other aspects of the learning such as selecting the candidate model for a particular neighborhood according to an information criterion or selecting the set of features to be considered in the model are handled by Weka (Hall et al. 2009) which is the tool that we use as the backbone linear regression learner in our methods.

Alg. 1 summarizes the process of training a committee of linear regression models trained by using an arbitrary neighborhood profile and computing the final target value based on the base model estimations. In Alg. 1, TS_d and m_d represent the d^{th} training set and the model built based on it respectively. Also, $CombineVals$ combines the estimations from individual models. In its simplest form $CombineVals$ returns the mean of the individual estimations as it is used in the experiments reported in the next section.

Algorithm 1 the generic stretched-based neighborhood selection ensemble method

Input:
q: input query
dim: number of dimensions
NP: neighborhood selection profile
Output: T: Estimated target value

```
for d ← 1 to dim do
    TS_d ← NeighborhoodSelection(NP, k, d)
    m_d ← Train(TS_d)
    ValEstimate_d ← Estimate(q, m_d)
end for
return CombineVals(∪_{d∈{1,...,dim}} ValEstimate_d)
```

Ensemble Selection for Gradient Method

Likewise the previous method, estimations from stretched neighborhoods are combined to form the final estimation. However, in the gradient method, the estimation for a single neighborhood is calculated by combining two components itself. In this case, one component is the estimation from a model trained by ascending trend inclined point selection and the other would be the estimation from the descending trend inclined point selection. Same criteria as explained for the previous method could be used for weighting each model in this case as well :

$$f(q) \equiv \sum_{i=1}^{n} \sum_{i=1}^{2} f_{i,j}(q) w_{i,j} \tag{7}$$

Here $f_{i,1}(q)$ and $f_{i,2}(q)$ are the estimations from a model trained by an ascending trend inclined and descending trend inclined point selection strategy respectively and $w_{i,j}$ is the weight assigned to each model.

Experimental Design

We conducted experiments on the four domains all from the UCI repository (Frank and Asuncion 2010): MPG, with 7 features and 392 cases, Auto Price, with 13 features and 195 cases, Housing, with 13 features and 506 cases, and Abalone, with 7 features and 4177 cases. Each data set is cleaned by removing cases with unknown feature values. For each input feature, values are standardized by subtracting that feature value's mean from each individual feature value and dividing the result by the standard deviation of that feature. Target values are not standardized.

Experiments assess the performance of different neighborhood selection strategies for training a linear regression model by comparing their mean absolute errors. The followings are the training data selection methods considered in our experiments:

1. Spherical-shaped: a radius-based (RB) neighborhood as the one depicted in part a of Fig. 1
2. Cylindrical-shaped (CS): a radius-based neighborhood with relaxing one of the dimensions as the one depicted in part b of Fig. 1 using the Euclidean version of Eq. 2.
3. Hourglass-shaped (HS): an hourglass-shaped neighborhood as depicted in part c of Fig. 1 achieved by using Eq. 3 and using $\frac{1}{|x_d - q_d|} + C$ in as K1.
4. Diamond-shaped (DS): a diamond-shaped neighborhood as the one depicted in part d of Fig. 1 using Eq. 3 and using $\mid x_d - q_d \mid$ in as K1.
5. Gradient-based (GB): a cylindrical-shaped neighborhood by using a kernel function like the one introduced in Fig. 2 for weighting the points in the selected neighborhood.

For each method, we tested for a sequence of parameter settings (e.g., settings the radius for the spherical neighborhood) to provide a sequence of neighborhoods of different sizes (e.g., the smallest neighborhood with 5 cases, smallest neighborhood with 10 cases, etc.), in order to compare performance with different neighborhood sizes. In all cases leave-one-out cross validation is used for evaluating the performance of each method.

Experimental Results

Spherical vs. Hourglass and Gradient-Based Methods

Fig. 3 depicts the percentage of improvement in terms of Mean Absolute Error (MAE) of hourglass-shaped and gradient-based training sample selection methods compared to the performance of the radius-based method, for different numbers of instances considered.

In all domains, for the same training set size, the hourglass and gradient-based methods outperform the spherical neighborhood method. This is especially true for smaller training sizes which acknowledges our hypothesis about the possible usefulness of using stretched neighborhoods for training locally weighted linear regression models. However, sometimes this gain is not noticeable. As an example, as depicted in part d of Fig. 3 for the Abalone domain, this improvement is negligible. On the other hand, for the Housing and

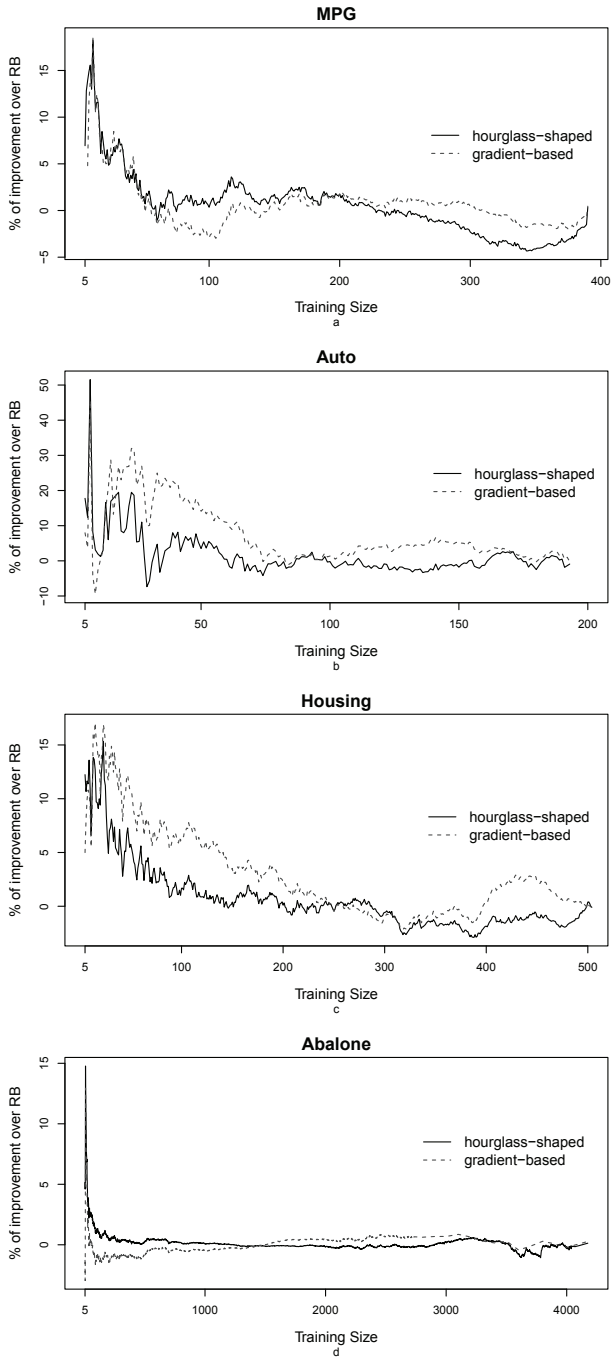

Figure 3: percentage of improvement in MAE for hourglass-shaped and gradient-based methods compared to that of radius-based training selection

Auto domains the gain is more significant. We hypothesize that this results because the instance sets in the Housing and Auto domains are relatively sparser compared to those in the Abalone domain. In the Abalone domain the average number of cases per solution is 149 while this value in Housing, Auto and MPG domains decreases to 2.21, 1.1 and 3.1 respectively. Also for neighborhoods that contain nearly the whole data set (i.e right end of graphs) for each domain all linear regression models are approximately built using the same training data and consequently hourglass and gradient-based methods show same performance as that of spherical-based method.

We observe that hourglass-shaped and gradient-based methods usually show very similar performance especially for Abalone and MPG domains. However, in general the gradient-based method shows better performance compared to hourglass-shaped method (especially in the Auto domain). This can justify the importance of trend detection for the target function and using that for guiding the training data selection over a stretched neighborhood.

Effectiveness of Training Data Selection Methods

Next, we consider all candidate training data selection methods introduced in the previous section and examine the maximum performance gain achievable from each. We use the best performance achieved by *k-NN* (with optimal k) as the baseline and compare the maximum gain in terms of the MAE from each candidate training data selection method compared to that of *k-NN*. Fig. 4 summarizes the percentage of improvement of each method compared to *k-NN*.

The range of improvement or deterioration varies in the different domains, with the Abalone domain showing the least variation in the percent of improvement (we hypothesize that this is for the same reasons explained for Fig. 3) and the Housing domain showing the greatest variation. While in the Abalone domain proposed methods (HS and GB) show less than 1% improvement (the actual MAE values for HS, GB and *k-NN* are 1.52, 1.52 and 1.53 respectively) this value is increased to 11% (MAE equals 1.80, 1.81 and 2.02), 7.5% (MAE equals to 1473, 1472 and 1589) and 15% (MAE equals 2.19, 2.12 and 2.5) for MPG, Auto and Housing domains respectively.

As depicted in parts b, c and d of Fig. 4, using a diamond-shaped neighborhood degrades the performance compared to k-NN in nearly all tested domains. Also, in most cases the hourglass and gradient-based methods show fairly close gains compared to k-NN, and always provide better performance than the maximum gain achieved by the spherical neighborhood method.

The cylindrical method yields results that are always better than spherical but worse than the hourglass neighborhood selection. This shows that using a method that is inclined to choose a wider range of training data, but still along the relaxed dimension, can be beneficial compared to using a cylindrical-shaped method. Also, although in Fig. 3 it is observed that the gradient-based method usually performs better than hourglass-shaped in all domains except Abalone for various training set sizes, Fig. 4 shows that for optimal configurations these two methods perform closer together.

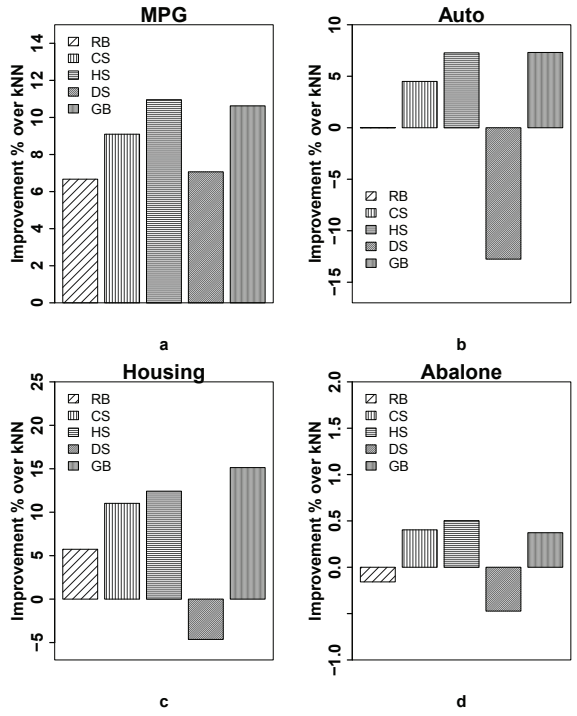

Figure 4: Percent of improvement in MAE of the best performance achieved by each candidate training data selection method compared to *k-NN*

For example, for the MPG and Abalone domains hourglass-shaped neighbor selection under the optimal configurations performs slightly better than the gradient-based method and for the Auto and Housing domain it is the gradient-based method that slightly performs better than the hourglass-shaped method.

Conclusion and Future Work

We have described new methods of selecting neighborhoods of an input query to use as training data for building linear regression models, and proposed ensemble methods for combining their results. The neighborhood selection methods stretch spherical neighborhoods in n-dimensional space along a single dimension at a time, generating n different models, whose values are then combined to generate the final target value. Experiments showed that using stretched neighborhoods for training linear regression models can improve accuracy measured by Mean Absolute Error compared to other candidate methods in four sample domains, especially if the space of target values is sparse.

Future work includes examining greedy techniques such as hill-climbing for selecting the optimal set of models trained by the stretched neighborhoods and their corresponding weights for calculating the final target value and examining the use of non-parametric regression estimators in combination with the stretched neighborhoods. Another future direction is exploring alternative non-linear and linear weighting functions for the gradient-based method.

References

Atkeson, C. G.; Moore, A. W.; and Schaal, S. 1997. Locally weighted learning. *Artif. Intell. Rev.* 11(1-5):11–73.

Breiman, L. 1996. Bagging predictors. *Mach. Learn.* 24(2):123–140.

Drucker, H. 1997. Improving regressors using boosting techniques. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, 107–115. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Frank, A., and Asuncion, A. 2010. UCI machine learning repository.

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

Friedman, J. H. 2000. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29:1189–1232.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl.* 11(1):10–18.

McSherry, D. 1998. An adaptation heuristic for case-based estimation. In *Proceedings of the 4th European Workshop on Advances in Case-Based Reasoning*, EWCBR '98, 184–195. London, UK, UK: Springer-Verlag.

Orr, M. J. L. 1996. Introduction to radial basis function networks.

Patterson, D.; Rooney, N.; and Galushka, M. 2002. A regression based adaptation strategy for case-based reasoning. In *Eighteenth national conference on Artificial intelligence*, 87–92. Menlo Park, CA, USA: American Association for Artificial Intelligence.

Schapire, R. E. 1990. The strength of weak learnability. *Mach. Learn.* 5(2):197–227.

Scholkopf, B., and Smola, A. J. 2001. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, USA: MIT Press.

Sun, Q., and Pfahringer, B. 2011. Bagging ensemble selection. In *Australasian Conference on Artificial Intelligence*, 251–260.

yau Kwok, T., and Yeung, D.-Y. 1997. Constructive algorithms for structure learning in feedforward neural networks for regression problems. *IEEE Transactions on Neural Networks* 8:630–645.

Ye, J.; Chow, J.-H.; Chen, J.; and Zheng, Z. 2009. Stochastic gradient boosted distributed decision trees. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, 2061–2064. New York, NY, USA: ACM.

Zhu, X., and Yang, Y. 2008. A lazy bagging approach to classification. *Pattern Recogn.* 41(10):2980–2992.

Zhu, X.; Bao, C.; and Qiu, W. 2008. Bagging very weak learners with lazy local learning. In *ICPR*, 1–4.