2010 IEEE International Conference on Data Mining

Passive Sampling for Regression

Hwanjo Yu and Sungchul Kim

{*hwanjoyu,subright*}*@postech.ac.kr*
Pohang University of Science and Technology (POSTECH)
Pohang, South Korea

***Abstract*—Active sampling (also called active learning or selective sampling) has been extensively researched for classification and rank learning methods, which is to select the most informative samples from unlabeled data such that, once the samples are labeled, the accuracy of the function learned from the samples is maximized. While active sampling methods require learning a function at each iteration to find the most informative samples, this paper proposes passive sampling techniques for regression, which find the informative samples not based on the learned function but based on the samples' geometric characteristics in the feature space. Passive sampling is more efficient than active sampling, as it does not require, at each iteration, learning and validating the regression functions and evaluating the unlabeled data using the function. For regression, passive sampling is also more effective; Active sampling for regression suffers from serious performance fluctuations in practice, because it selects the samples of highest regression errors and such samples are likely noisy. Passive sampling, on the other hand, shows more stable performance. We observe from our extensive experiments that our passive sampling methods perform even better than the "omniscient" active sampling that knows the labels of unlabeled data.**

***Keywords*-Selective sampling, Passive sampling, Active learning, Active sampling, Regression**

I. Introduction

In most supervised learning problems (*i.e.*, classification, regression, and ranking problem), a training set is assumed to be labeled. However, labeled data is often difficult, time-consuming, or expensive to obtain in many domains. In such domains, it is desirable to reduce the cost of labeling training data while maintaining the quality of the learned function. For example, in an automatic movie scoring system, we can ask users to score movies (*i.e.*, labels of data), from which a regression function can be learned to predict scores of new movies. Collecting a large set of labeled data is costly in this case but is also important to guarantee inducing an accurate regression function.

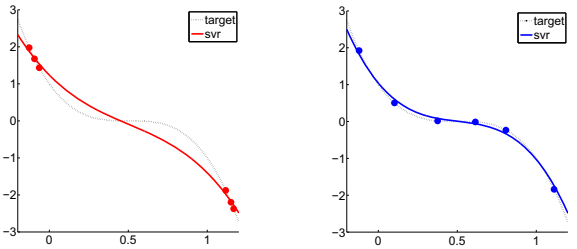
A solution called active sampling (or called active learning or selective sampling) is to select the most informative samples from the unlabeled data such that, once the samples are labeled, the accuracy of the function learned from the sample is maximized. The typical framework for active sampling starts from a small set of labeled data and a large set of unlabeled data: First, a function is learned from the small set of labeled data. The system then, using the function, selects another small informative sample from the pool of unlabeled data. After that, the sample is labeled by users, and the system learns a function again with the accumulated labeled data. This process of learning and sampling is iterated until the learned function becomes accurate enough. Active sampling techniques are used when the system selects the most informative sample in order to minimize the number of iterations to achieve a high accuracy.

The active sampling has been extensively researched for classification or rank learning methods (such as for SVM or RankSVM) [9], [10], [1], [11], [3], [6]. The key ideas of most active sampling techniques is to select the most ambiguous or uncertain sample for classification or ranking. The most uncertain sample can be determined based on the function learned at the previous iteration; For example, the sample nearest to the classification boundary can be considered most uncertain for classification [9], and the data pairs whose ranking distance is the shortest can be considered most uncertain for ranking [11]. The distance to the classification boundary or the ranking distance between data pairs can be computed using the function learned at the previous iteration.

The active sampling for regression method (such as SVR (Support Vector Regression)) is more challenging because, in regression, it is nontrivial to identify the uncertainties of data points based on the function learned at the previous iteration [2]; Regression is to estimate the target value (represented as a real number $y \in \mathcal{R}$) of an input object (often represented as a vector $\mathbf{x}$). The uncertainty of a data point in regression is computed by the regression error, *i.e.*, $||f(\mathbf{x}_i) - y_i||$, but y_i is unknown for unlabeled data $\mathbf{x}_i$, thus the regression error is hard to estimate for unlabeled data.

This paper proposes *passive sampling* techniques for regression, that is to select the sample *a priori* before learning. Passive sampling is distinguished from traditional active sampling in that, while active sampling selects the sample based on the function learned at the previous round, the sample in passive sampling is selected based on their geometric characteristics in the feature space, thus no learning is required before sampling. In other words, the uncertainties of data points are determined based on their geometric characteristics in the feature space rather than their regres-

1550-4786/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDM.2010.9



(a) SVR with skewed sample (b) SVR with unskewed sample

Figure 1. Example of SVR functions trained from six skewed sample and unskewed sample.

sion errors. Passive sampling is more efficient than active sampling, because passive sampling does not require, at each iteration, learning and validating the regression function and evaluating the unlabeled data using the function.

This paper propositions that, for regression, passive sampling is also more accurate and stable than active sampling. While active sampling performs well in noiseless data sets, in practice, real data sets are not exactly generated from a regression function. Active sampling for regression selects the samples of highest regression errors and such samples are likely noisy. Thus, we find that active sampling suffers from serious performance fluctuations on real-world data. Our experiment results show that our passive sampling methods perform even better than the "omniscient" active sampling that knows the labels of unlabeled data.

This paper is organized as follows. Section II presents the objective of passive sampling and presents passive sampling heuristics to optimize the objective. Section III reports the experimental results. Section IV concludes our study.

II. Passive Sampling for Regression

This section first discusses the objective of passive sampling for regression (Section II-A) and presents the passive sampling methods (Section II-B).

A. The Objective of Passive Sampling

To motivate our passive sampling heuristics for regression, we illustrate an example:

Example 1: Figure 1 shows regression curves learned from the SVR. Both the red curve in the left figure and the blue curve in the right figure are trained from six data points. However, the blue curve in right figure is drawn more closely to the target function. The main difference between two figures is that the data points in the right figure are well spreaded while the points in the left figure are skewed. Intuitively, when the data is skewed, the area with scarce data points in the feature space will not have enough data to draw an accurate regression function.

Based on the intuition, we compute the uncertainty $u(\mathbf{x})$ of a point $\mathbf{x}$ in a sample S based on its distance to the nearest neighbor in the sample, because the regression function is likely uncertain in the area of around x when there is no points around $\mathbf{x}$:

$$u(\mathbf{x}, S) = \min_{\forall \mathbf{x}_i \in S \setminus \mathbf{x}} dist(\mathbf{x}, \mathbf{x}_i) \tag{1}$$

where $dist()$ is a distance in the feature space. The uncertainty of $\mathbf{x}$ according to Eq.(1) is computed by the distance between $\mathbf{x}$ and its nearest neighbor. For example, the uncertainty of $\mathbf{x}$ increases, as its nearest neighbor becomes farther.

The goal of our passive sampling is thus to select the sample S from the data set D such that the data points within S have the highest uncertainties as follows.

$$S = \arg\max_{\forall S \subset D} \left(\sum_{\forall \mathbf{x} \in S} u(\mathbf{x}, S) \right) \tag{2}$$

That is, the sample S that optimize Eq.(2) is the most informative sample for regression. A naive way of finding S that optimizes Eq.(2) will take an exponential time, as there is a combinatorial number of S in D.

B. Passive Sampling Heuristics

This section first discusses two straightforward heuristics but not practically viable – Grid and k-center algorithms, and then presents two reasonable and viable heuristics – greedy algorithm and incremental k-medoids.

1) Grid approach: Since the objective is improved by finding S uniformly distributed in the feature space, a grid approach is a possible solution, which divides each dimension into equal-width bins and picks data points from each cell. However, the number of cells increases exponentially with the number of dimensions while the size of S is often small. For example, suppose we pick ten data points $|S| = 10$ and the dimension is ten. Then, even if we divide each dimension only into two bins, the number of cells becomes $2^{10} = 1024$ while we only need to pick ten data points. Thus, the grid approach will be highly inefficient and easily fail to sample in a uniformly distributed way.

2) k-center algorithms: The problem of finding the sample optimizing the objective function in Eq.(2) is similar to the k-center problem, as it is also to find the k points in a uniformly distributed way. The k-center problem is originally to find the k facilities which minimize the maximal distance between a demand point and its closest facility. Let $V = \{v_1, \cdots, v_n\}$ be a set of n candidate facilities, and $U = \{u_1, \cdots, u_m\}$, a set of m demand points. The distance between a demand point and its closest facility (v_i, u_j) is given as $d(u_i, v_j)$. Then, the k-center problem is to find a subset $X \in V$ of size k such that

$$\max_{u_i \in U} \min_{v_j \in X} d(u_i, v_j) \tag{3}$$

The k-center problem is also known to be NP-hard. Approximation algorithms were developed such as Vertex-Interchange Method [5], Chain-Interchange Tabu Search Method [4], [8], and Dominanting Set Algorithm [7] However, these algorithms are not directly applicable, as they do not find k "incrementally". Passive sampling accumulates the sample found at each iteration to the training set for regression. That is, the samples selected previously cannot be unselected, and the sum of uncertainties for both the newly selected sample and previously selected samples must be maximized. In other words, the objective function for new sample is affected by the previously selected samples.

3) Greedy algorithm: We can select new samples in a greedy way such that the new sample is located far from the previously selected and labeled samples. Specifically, for the sets of labeled data X and unlabeled data Z, we compute the minimum distance between each unlabeled data point $\mathbf{z}$ and X, and pick the $\mathbf{z}$ having the largest minimum distance as follows.

$$\arg\max_{\mathbf{z}(\in Z)} \left(\min_{\forall \mathbf{x} \in X} dist(\mathbf{x}, \mathbf{z}) \right) \tag{4}$$

The greedy algorithm is described in Algorithm 1. Note that the greedy algorithm is incremental; the algorithm can be run repeatedly by moving the selected points from Z to S at each iteration. The algorithm has the complexity of $O(|X||Z|)$ at the first iteration and $O(k|Z|)$ from the second iterations, because the distances computed at the previous iterations can be reused in the following iterations.

Algorithm 1 Greedy sampling

Input: labeled data set X, unlabeled data set Z
Output: sample S of size k.

1: $S = \emptyset$
2: **while** $|S| < k$ **do**
3: Calculate the minimum distance between each unlabeled data point $\mathbf{z}$ and $\{S \cup X\}$.
4: Select the $\mathbf{z}$ that has the largest minimum distance.
5: Move $\mathbf{z}$ from Z to S.
6: **end while**

4) Incremental k-medoids algorithm: We revise the k-medoids algorithm to an incremental version to find the sample of k points that approximately optimizes our objective. The k-medoids algorithm is similar to the k-mean clustering algorithm except that, in k-medoids, the cluster centers are selected from the data while the centers are the computed centroids in the k-mean.

In order to make it incremental and optimizing our objective, the sample of k data points must be selected far from the labeled data and previously selected points. Thus, we initially select the sample using the greedy algorithm. We then assign the unlabeled data points to the nearest labeled or selected sample and run the k-medoids algorithm with only the unlabeled points assigned to the selected sample. While running the k-medoids, the initially selected sample of k points will be updated expecting to improving the objective. Algorithm 2 describes the incremental k-medoids algorithm.

Algorithm 2 Incremental K-medoids sampling

Input: labeled data set X, unlabeled data set Z
Output: sample S of size k.

1: Select S from Z using Algorithm 1.
2: // $Z = Z - S$
3: **for** each $\mathbf{z}$ in Z **do**
4: Assign $\mathbf{z}$ to it's closest point in $\{S \cup X\}$.
5: **end for**
6: Let Z' be the set of $\mathbf{z}$ assigned to S.
7: $d^* =$ The sum of distances between $\mathbf{z}' \in Z'$ and its point in S.
8: **for** each point $\mathbf{s}$ in S **do**
9: **for** each unlabeled point $\mathbf{z}'$ in Z' **do**
10: $d =$ The sum of distances between $\mathbf{z}' \in Z'$ and its point in S after swaping $\mathbf{s}$ and $\mathbf{z}'$
11: **if** $d < d^*$ **then**
12: Swap $\mathbf{s}$ and $\mathbf{z}'$
13: $d^* = d$
14: **end if**
15: **end for**
16: **end for**
17: Repeat step 3 to 16 until there is no swaping between S and Z'

III. Experiments

This section evaluates the passive sampling heuristics – (1) the greedy (Greedy) and (2) the incremental k-medoids sampling (KMedoid) – against (3) random sampling (Random) and (4) "omniscient" active sampling (Active) assuming the system knows the labels of unlabeled data. Note that the omniscient method is kind of "cheating", as the labels are unknown in practice.

For evaluation metric, we use the mean absolute error (MAE), *i.e.*, $\frac{1}{|T|}\sum |y - f(\mathbf{x})|$, between the target outputs y and regression outputs $f(\mathbf{x})$ on the testing set T. We divided each data set into training and testing set, and the training samples are only chosen from the training set. We used the LIBSVM for the SVR implementation.

A. Synthetic Data

We generated two sets of synthetic data – noiseless and noisy. Each set is composed of randomly generated 1000 vectors of 10 dimensions, some (80%) of which are used for training and the others (20%) for testing. We then randomly generated a weight vector $\mathbf{w}$, and the target regression function $f^*(\mathbf{x})$ is a parametric function of $\mathbf{w}$ as follows.

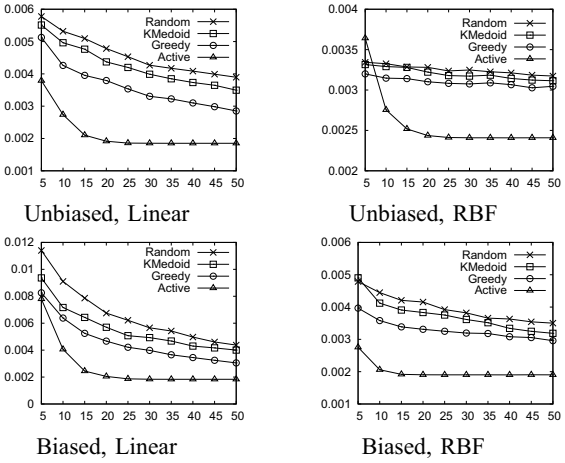


Figure 2. MAE on *noiseless* data. X-axis: # of instances; Y-axis: MAE; "Unbiased": unbiased labeled set; "Biased": biased labeled set; "Linear": linear kernel; "RBF": RBF kernel.

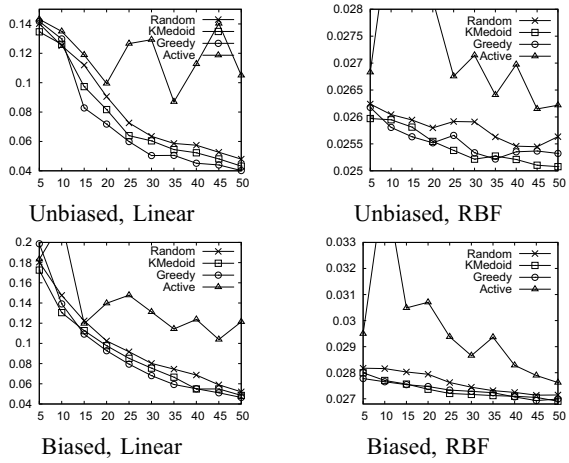


Figure 3. MAE on *noisy* data. X-axis: # of instances; Y-axis: MAE; "Unbiased": unbiased labeled set; "Biased": biased labeled set; "Linear": linear kernel; "RBF": RBF kernel.

$$\text{linear function:} \quad f^*(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

$$\text{RBF function:} \quad f^*(\mathbf{x}) = exp\left(-\frac{||\mathbf{x}-\mathbf{w}||^2}{2\sigma^2}\right)$$

For each of the target regression function, we generated the target output y by evaluating $f^*(\mathbf{x})$, to generate the noiseless data set. (We fixed the parameter $\sigma = 1$.) For the noisy data set, we added small noise ($\sim N(0, 0.1)$) to y of 20% of data.

We first sampled two sets of 50 labeled points (1) randomly and (2) with a bias. Note that the labeled set can be easily biased in practice. We experimented with these two sets of labeled data independently. For the biased set, we pick a random point and pick the other 49 points from its neighbors only. Starting from the labeled set of 50 points, we selected 5 points ($|S| = 5$) at each iteration using each of the four sampling methods. We then learned regression functions using the SVR with linear and RBF kernels (linear kernel for the linear function and RBF kernel for the RBF function) and evaluated them on the testing set. We fixed $\sigma = 1$ and tuned the soft margin parameter $C = 10^{-3}, ..., 10^3$.

Figure 2 shows the MAEs of the sampling methods at each iteration on the noiseless data set when the labeled set is unbiased and biased respectively. The results are averaged over 30 runs. The passive sampling methods Greedy and KMedoid perform worse than Active but better than Random. *In the noiseless synthetic data,* Active *performs the best.* However, as Figure 3 show, *when the data sets are noisy, the passive sampling methods perform better than the others,* Active *suffers from serious accuracy fluctuation by focusing on noisy samples.* Our experiments on real data sets in Section III-B also show similar results.

Name	# instances	# attributes	Source
Concrete	1030	7	UCI
Cpusmall	8192	12	Statlib
Housing	506	13	UCI
Mg	1385	6	Statlib
Mpg	392	7	UCI
NO2	500	7	Statlib
Places	329	9	Statlib
PM10	500	7	Statlib
Space_ga	3107	6	Statlib

Table I
DATA SETS

B. Real Data

This section performs experiments on real data sets of various sizes that we acquired from the UCI machine learning repository and the Statlib from CMU as described in Table I. We used the RBF kernel for this experiment and tuned the soft margin parameters C and the kernel parameter γ from 10^{-3} to 10^3 and reported the best accuracy. The results are also averaged over 30 runs.

Figure 4 and 5 compare the accuracy of sampling methods on the real data sets, with unbiased and biased labeled sets respectively. In both figures, *the passive sampling algorithms,* Greedy *and* KMedoid *almost always perform better than* Random *and* Active*, and* Greedy *and* KMedoid *are comparable.* Note that, Active fluctuates significantly especially when the labeled set is unbiased. It is because Active focuses on sampling the points of highest errors and such samples are likely noisy in real data sets.

1) Objective scores: Table II compares the objective scores of the sampling methods after sampling 50 data points. KMedoid and Greedy always generated higher objective scores than Active and Random.

2) Sampling time: Table III compares the sampling time of each method when sampling the next 5 points after

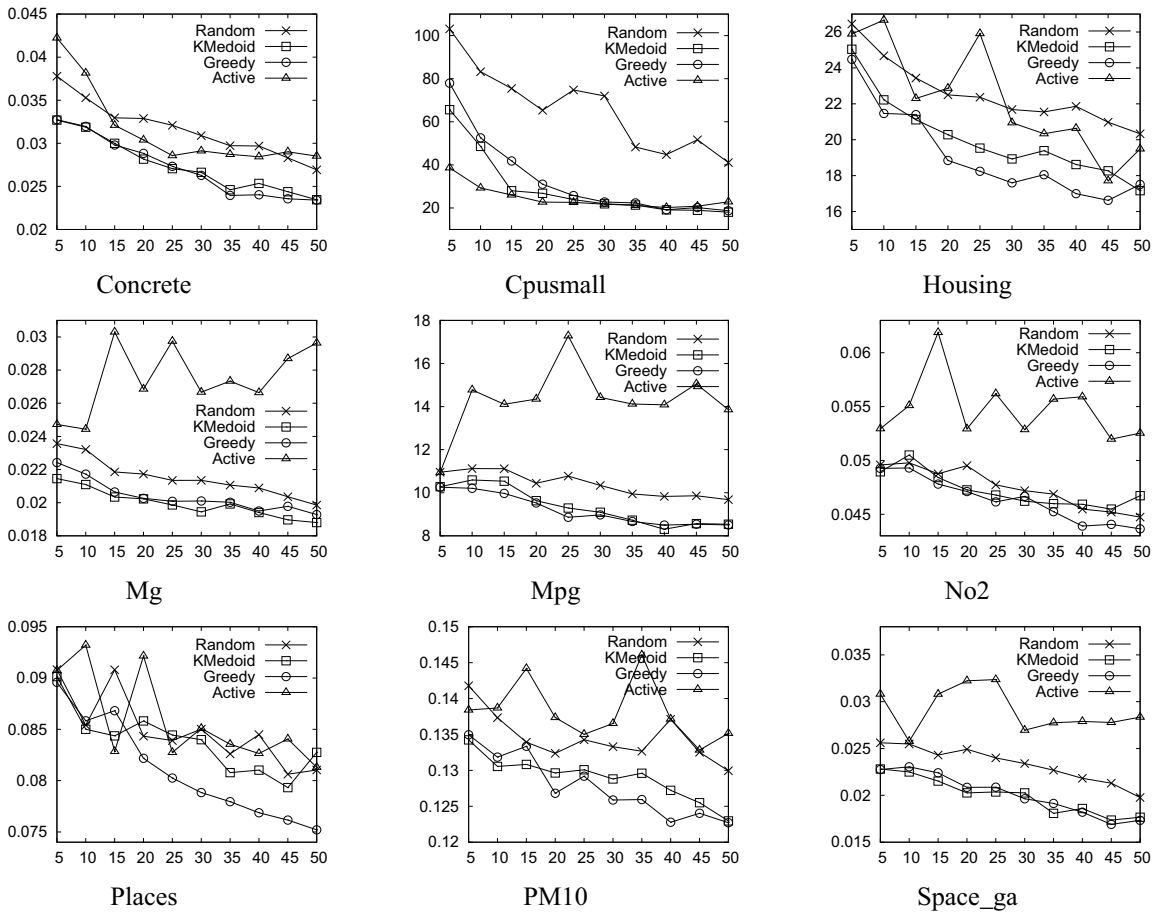


Figure 4. MAE on real data sets with the unbiased labeled set.

Dataset	Random	KMedoid	Greedy	Active
Concrete	36.92	50.79	**60.37**	29.46
Cpusmall	19.99	43.63	**55.86**	30.83
Housing	46.13	63.26	**70.16**	47.58
Mg	21.88	32.10	**36.25**	17.29
Mpg	29.40	38.49	**42.38**	29.38
No2	45.98	54.85	**62.56**	46.66
Places	40.92	**54.05**	53.15	42.90
Pm10	44.35	53.76	**62.06**	46.07
Space_ga	15.68	25.07	**31.27**	17.05

Table II
THE OBJECTIVE SCORES IN EQ. (2) OF EACH SAMPLING METHOD AFTER SAMPLING 50 DATA POINTS

Dataset	Random	KMedoid	Greedy	Active
Concrete	0	0.06	**0.01**	0.28
Cpusmall	0	0.795	**0.105**	0.445
Housing	0	0.025	**0.0**	0.45
Mg	0	0.09	**0.02**	0.575
Mpg	0	0.01	**0.005**	0.585
No2	0	0.02	**0.0**	0.445
Places	0	0.02	**0.0**	0.785
Pm10	0	0.02	**0.0**	0.565
Space_ga	0	0.225	**0.03**	0.805

Table III
THE TIME TO SAMPLE THE NEXT 5 POINTS AFTER SAMPLING 45 POINTS

sampling 45 points. As expected, Random is the fastest and Greedy is the next. Active is the slowest, as it includes the processes of learning and validating regression functions.

IV. CONCLUSIONS

This paper proposes *passive sampling techniques* for regression. Unlike active sampling, passive sampling does not require learning a function before sampling. Passive sampling instead computes the uncertainties of data points based on their geometric characteristics and selects the points of highest uncertainties. This paper shows that, for regression, passive sampling is a better choice than active sampling. Not only it is more efficient, but also it is more accurate; Since regression data is likely noisy in practice, active sampling incurs the regression function easily overfitting to the noisy training data, thus its performance suffers from serious fluctuations.

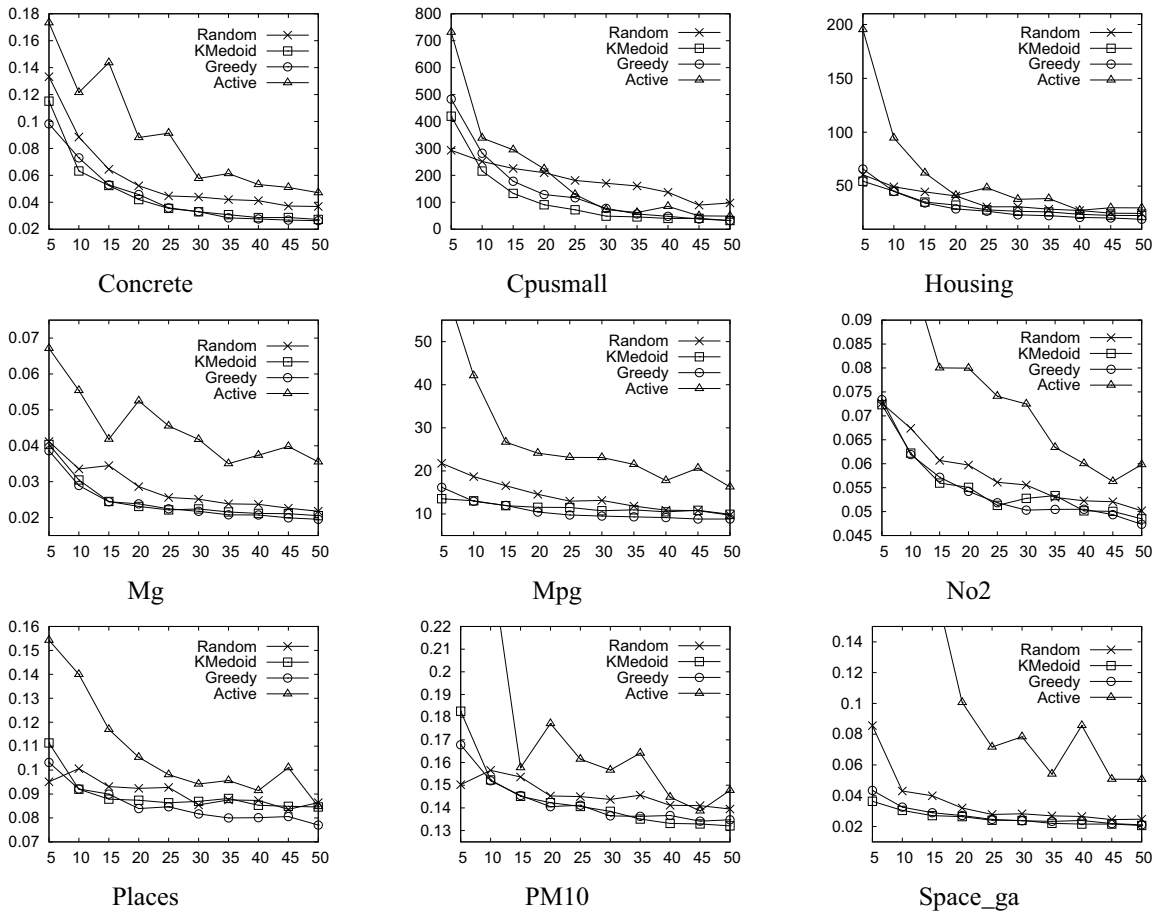


Figure 5. MAE on real data sets with the biased labeled set.

ACKNOWLEDGMENT

This work was supported by the Brain Korea 21 Project in 2010 and Mid-career Researcher Program through NRF grant funded by the MEST (No. KRF-2009-0080667).

REFERENCES

[1] K. Brinker. Active learning of label ranking functions. In *Proc. Int. Conf. Machine Learning (ICML'04)*, 2004.

[2] R. Castro, R. Willett, and R. Nowak. Faster rates in regression via active learning. In *Proc. Advances in Neural Information Processing Systems (NIPS'05)*, 2005.

[3] E. Chang and S. Tong. Support vector machine active learning for image retrieval. In *ACM Int. Conf. Multimedia (MM'01)*, 2001.

[4] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computer Operational Research*, 1986.

[5] P. Hansen and N. Mladenovi. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 2001.

[6] B. Long, O. Chapelle, Y. Zhang, Y. Chang, Z. Zheng, and B. Tseng. Active learning for ranking through expected loss optimization. In *Proc. ACM SIGIR Int. Conf. Information Retrieval (SIGIR'10)*, 2010.

[7] J. Mihelic and B. Robic. Solving the k-center problem efficiently with a dominanting set algorithm. *Journal of Computing and Information Technology*, 2005.

[8] K. Rosing, C. ReVelle, E. Rolland, D. Schilling, and J. Current. Heuristic concentration and tabu search: A head to head comparison. *European Journal of Operational Research*, 1998.

[9] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proc. Int. Conf. Machine Learning (ICML'00)*, pages 839–846, 2000.

[10] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proc. Int. Conf. Machine Learning (ICML'00)*, pages 999–1006, 2000.

[11] H. Yu. SVM selective sampling for ranking with application to data retrieval. In *Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD'05)*, 2005.