Kernel K-means Based Framework for Aggregate Outputs Classification

Shuo Chen, Bin Liu, Mingjie Qian, Changshui Zhang
State Key Laboratory on Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Automation, Tsinghua University, Beijing 100084, China
{chenshuo07, liubin07} @mails.tsinghua.edu.cn
qian.mingjie@gmail.com, zcs@mail.tsinghua.edu.cn

***Abstract*—Aggregate outputs learning is a newly proposed setting in data mining and machine learning. It differs from the classical supervised learning setting in that, training samples are packed into bags with only the aggregate outputs (labels for classification or real values for regression) provided. This problem is associated with several kinds of application background. We focus on the aggregate outputs classification problem in this paper, and set up a framework based on kernel K-means to solve it. Two concrete algorithms based on our framework are proposed, each of which can cope with both binary and multi-class scenarios. The experimental results suggest that our algorithms outperform the state-of-art technique. Also, we propose a new setting for patch extraction in the Content Based Image Retrieval procedure by using the algorithm.**

***Keywords*-Kernel K-means; Aggregate Outputs Learning; Content Based Image Retrieval;**

Bag ID	Height	Weight	Aggregated Label
1	168cm	62kg	3 F, 2 M
	165cm	57kg	
	157cm	53kg	
	176cm	69kg	
	154cm	46kg	
2	171cm	58kg	1 F, 2 M
	192cm	101kg	
	163cm	66kg	

Table I
AN EXAMPLE OF TRAINING DATASET FOR AGGREGATE OUTPUTS CLASSIFICATION. THERE ARE 8 INSTANCES, EACH WITH TWO FEATURES OF HEIGHT AND WEIGHT OF A PERSON. THE INSTANCES ARE PACKED INTO 2 BAGS. THE AGGREGATED LABELS SHOW HOW MANY MALES AND FEMALES IN EACH BAG.

I. INTRODUCTION

The problem of aggregate outputs learning (AOL) was first proposed by D. Musicant *et al.* in their original work [1]. It is raised from the application in analyzing the single particle mass spectrometry (SPMS) data. It is expected to train a predictor of the quantity of *black carbon* (BC) for any given single particle. However, the best possible instrument can only render imprecise measuring results in the form of aggregated BC (the sum of BC) for bags of training particles. Thus, one cannot use the mature supervised learning techniques to solve this problem directly. The SPMS problem is of regression nature. It can also be extended to the classification scenario as the authors did in [1], which is what we are concerned about in this paper. It is different from classical supervised classification in that, the training samples are packed into several disjunct bags with only aggregated labels given, which suggest the numbers of samples from different classes in each bag. We show an example of training dataset for this setting in Table I.

The literature of AOL is scarce. The original work [1] is the only one that strictly follows the setting mentioned above to the best of our knowledge. In [1], three classical supervised learning algorithms were adapted into the AOL scenario, namely k-nearest-neighbor (kNN), artificial neural network (ANN) and support vector machine (SVM). The authors provided both classification and regression versions of the algorithm. In this paper, we only focus on solving the aggregate outputs classification (AOC) problem, and we denote the three classification algorithms as AOC-kNN, AOC-ANN and AOC-SVM respectively.

In this paper, we take the AOC problem as a constrained clustering problem. We set up a framework based on kernel K-means ([2]) to reveal the hidden label of each sample. Then we can use them to train a supervised classifier to induct on unseen samples. We conduct experiments on two datasets, and the results are promising. We also propose a new method of patch extraction for Content Based Image Retrieval (CBIR) using AOC algorithm, with satisfying experimental results shown.

II. PROBLEM FORMULATION

Our formulation is slightly different from that of [1] just for convenience in mathematical derivation. They describe the same problem actually.

Suppose we are provided with a training dataset with n samples $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ belonging to c classes. $\mathbf{x}_i$ is the d-dimensional column vector that represents the ith sample, and each of the samples $\mathbf{x}_i$ is from the same input space χ with $\chi \in \mathbb{R}^d$. We also assume that each sample $\mathbf{x}_i$ is associated with a hidden y_i that takes the value of $1, 2, \ldots, c$. However, these labels cannot be seen directly. The n samples have been aggregated into m disjunct bags, with an $m \times n$ aggregating matrix A provided. The element of A is given

Algorithm 1: K-means clustering algorithm

Input: The dataset for clustering $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, the number of clusters c;

Output: $c \times n$ 0-1 membership matrix T;

Initialize T randomly or according to certain rules;

repeat

$$\mathbf{m}_i = \frac{\sum_{j=1}^{n} T_{ij}\mathbf{x}_j}{\sum_{j=1}^{n} T_{ij}}, i = 1, 2, \ldots, c; \quad (2)$$

for $i = 1 : n$ **do**

if

$$\|\mathbf{x}_i - \mathbf{m}_k\|_2^2 \leq \|\mathbf{x}_i - \mathbf{m}_j\|_2^2, \forall j = 1, 2, \ldots, c; \quad (3)$$

then

$T_{ki} = 1$;

$T_{li} = 0, \forall l \neq k$;

end

end

until T *converges* ;

according to

$$A_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_j \text{ belongs to the } i\text{th bag} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

We are also provided with an $m \times c$ matrix B for the aggregated labels, with its element B_{ij} equaling the number of the samples in the ith bag that belongs to the jth class.

Our goals for solving the AOC problem can be both transductive and inductive. For the transductive goal, we expect to give the label of each $\mathbf{x}_i$ that is as close to the hidden label as possible. Very often, our inductive goal is simplified to get a good predictive accuracy on a test dataset with known labels that is from the same distribution as that of the training set.

III. Review of K-means and Kernel K-means

The K-means clustering algorithm ([3]) is very popular in the whole literature of pattern recognition, machine learning and data mining due to its simplicity and effectiveness. It calculates the center for each cluster and assign the samples to the nearest cluster represented by the center repeatedly. We list it in Alg. 1. For the final result, the label $\mathbf{x}_i$ is j if and only if $T_{ji} = 1$.

The essence of the K-means is a greedy algorithm to solve following optimization problem

$$\min_T \sum_{i=1}^{c} \sum_{j=1}^{n} T_{ij} \|\mathbf{x}_j - \mathbf{m}_i\|_2^2 \quad (4)$$

subject to

$$\mathbf{m}_i = \frac{\sum_{j=1}^{n} T_{ij}\mathbf{x}_j}{\sum_{j=1}^{n} T_{ij}}, i = 1, 2, \ldots, c \quad (5)$$

$$T_{ij} = 0 \text{ or } 1, \forall \text{ feasible integers } i \text{ and } j \quad (6)$$

$$\sum_{i=1}^{c} T_{ij} = 1, \forall j = 1, 2, \ldots, n \quad (7)$$

One can also take K-means as an instance of the well-known Expectation-Maximization (EM) algorithm ([4]). In this view, we can see that K-means is based on the implicit assumption that the samples for clustering are from a same Gaussian mixture distribution, and each mixture corresponds one cluster. Thus K-means will not work well if this assumption is not satisfied in the original space χ. As a result, we need to nonlinearly map the samples in the original space to a high-dimensional one where K-means can work better sometimes. We denote this nonlinear mapping function as $\phi(\mathbf{x})$. However, rather than mapping directly with $\phi(\mathbf{x})$, we make use of the kernel function

$$K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle \quad (8)$$

K need to be a Mercer kernel to ensure that it can be a legitimate inner product function in the high-dimensional space. One can refer to [5] for more details.

By utilizing the kernel trick, we obtain the kernel K-means ([2]) algorithm. The difference with original K-means is that, we can no longer write the explicit form of the center of the cluster in high-dimensional space. Thus the step of (2) should be removed, and the calculation of distances in (3) becomes

$$\begin{aligned} \|\phi(\mathbf{x}_i) - \mathbf{m}_k\|_2^2 &= \left\| \phi(\mathbf{x}_i) - \frac{\sum_{j=1}^{n} T_{kj}\phi(\mathbf{x}_j)}{\sum_{j=1}^{n} T_{kj}} \right\|_2^2 \\ &= K(\mathbf{x}_i, \mathbf{x}_i) - 2\frac{\sum_{j=1}^{n} T_{kj} K(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{j=1}^{n} T_{kj}} \\ &\quad + \frac{\sum_{j=1}^{n}\sum_{j'=1}^{n} T_{kj}T_{kj'}K(\mathbf{x}_j, \mathbf{x}_{j'})}{\sum_{j=1}^{n}\sum_{j'=1}^{n} T_{kj}T_{kj'}} \end{aligned} \quad (9)$$

The other steps of the kernel K-means algorithm will remain the same as original K-means.

IV. Kernel K-means for AOC

In this section, we introduce our framework based on kernel K-means for the AOC problem.

A. Main Framework

We propose to solve the following optimization function

$$\min_T \sum_{i=1}^{c} \sum_{j=1}^{n} T_{ij} D_{ij} + \lambda L(A, B, T) \quad (10)$$

subject to

$$\begin{aligned} D_{ij} &= \|\phi(\mathbf{x}_j) - \mathbf{m}_i\|_2^2 \\ &= K(\mathbf{x}_j, \mathbf{x}_j) - 2\frac{\sum_{k=1}^{n} T_{ik} K(\mathbf{x}_j, \mathbf{x}_k)}{\sum_{k=1}^{n} T_{ik}} \\ &\quad + \frac{\sum_{k=1}^{n}\sum_{k'=1}^{n} T_{ik}T_{ik'}K(\mathbf{x}_k, \mathbf{x}_{k'})}{\sum_{k=1}^{n}\sum_{k'=1}^{n} T_{ik}T_{ik'}} \end{aligned} \quad (11)$$

$$T_{ij} = 0 \text{ or } 1, \forall \text{ feasible integers } i \text{ and } j \quad (12)$$

$$\sum_{i=1}^{c} T_{ij} = 1, \forall j = 1, 2, \ldots, n \tag{13}$$

where D is a $c \times n$ square distance matrix between samples and centers of clusters. Its calculation follows that in (9).

The first term of our objective function in (10) is the same with the original objective function of kernel K-means, which differs from that of K-means in (4) only by using $\phi(\mathbf{x}_i)$ rather than $\mathbf{x}_i$. $L(A, B, T)$ in the second term is a loss function concerning the aggregated labels. We elaborate it in the next subsection. λ is a tunable parameter which controls the tradeoff between the two terms.

B. Loss Function for Aggregated Labels

Let us consider the ideal aggregating case where the learned partition indicated by T fits the bag constraints (which can be thought as prior) completely. In this case, we can write

$$AT^T = B \tag{14}$$

However, it cannot be always this ideal in application problems due to the noise of the aggregated labels, and we need to make our algorithm flexible. As a result, rather than using hard constraints to ensure the bag prior, we add the second term of loss function to the objective function in (10). For the convenience in solving the later optimization problems, we use two convex functions in this paper

$$L_1(A, B, T) = \sum_{i=1}^{m} \sum_{j=1}^{c} U_\epsilon((AT^T)_{ij} - B_{ij}) \tag{15}$$

$$L_2(A, B, T) = \frac{1}{2} tr\Big((AT^T - B)^T (AT^T - B)\Big) \tag{16}$$

where

$$U_\epsilon(t) = H_{-\epsilon}(t) + H_{-\epsilon}(-t) \tag{17}$$

$$H_\theta(t) = \max(0, \theta - t) \tag{18}$$

$H_\theta(t)$ is the frequently used hinge loss function and $L_1(A, B, T)$ can be taken as a matrix version of double sided hinge loss function. $L_2(A, B, T)$ is a version of quadratic loss. We denote our algorithms using L_1 and L_2 as AOC-KKmeans-L1 and AOC-KKmeans-L2 respectively.

C. Solution of the Optimization Problem

Although L_1 and L_2 are both convex with respect to continuous T, the constraint in (12) requires T to be 0-1 so that it can reveal the partition. Thus we cannot utilize the well-developed convex optimization tools to solve the optimization as a whole. We propose an EM-like alternative updating algorithm for the solution. This kind of technique is guaranteed to converge to a local minimum ([4]). For each iteration, we first calculate the square distance matrix D according to the current membership indicator T. Then we solve an sub-optimization problem to assign each sample to a cluster. (We discuss the sub-optimization problems for AOC-KKmeans-L1 and AOC-KKmeans-L2 in the remainder of this subsection.) These procedures are repeated until convergence. We list the details of the algorithms in Alg. 2. It is worth mentioning that the initialization for T using pseudo-inverse is very important. It gives a good starting point so that the algorithm converges quickly. Also, it yields better solution than using other methods for initialization such as randomly generation according to our experiments.

The sub-optimization problem for AOC-KKmeans-L1 when D fixed can be rewritten into the following linear programming

$$\min_T \sum_{i=1}^{c} \sum_{j=1}^{n} T_{ij} D_{ij} + \sum_{i=1}^{m} \sum_{j=1}^{c} \eta_{ij} \tag{19}$$

subject to

$$\eta_{ij} \geq 0 \tag{20}$$

$$-\epsilon - \eta_{ij} \leq (AT^T - B)_{ij} \leq \epsilon + \eta_{ij} \tag{21}$$

$$0 \leq T_{ij} \leq 1 \tag{22}$$

$$\sum_{i=1}^{c} T_{ij} = 1, \forall j = 1, 2, \ldots, n \tag{23}$$

We have omitted some range specification for the subscripts in the constraints for tidiness. One should note that we use the constraint (22) rather than (12). The following theorem ensures the equivalence.

Theorem 1. *The optimal T_{ij} of linear programming (19)-(23) is* 0 *or* 1.

We omit the proof due to the lack of space. By using Theorem 1, we can solve the linear programming efficiently instead of the original mixed integer programming, whose time complexity is of NP-hard for the worst case. Also we do not need to restrict T_{ij} in the initialization step to be 0-1, since the following iterations will naturally enforce this constraint. For standard form of linear programming, we use the vectorization function $vec(\cdot)$ (along columns), $rvec(\cdot)$ (along rows) and Kronecker product $\otimes$ to write the matrix form

$$\min_{rvec(T), vec(\eta)} rvec(D)^T rvec(T) + \mathbf{1}_{m \times c}^T vec(\eta) \tag{24}$$

subject to

$$vec(\eta) \geq 0 \tag{25}$$

$$-\epsilon \mathbf{1}_{m \times c} - vec(\eta) \leq (I_c \otimes A) rvec(T) - vec(B) \leq \epsilon \mathbf{1}_{m \times c} + vec(\eta) \tag{26}$$

$$0 \leq rvec(T) \leq 1 \tag{27}$$

$$(\mathbf{1}_c^T \otimes I_n) rvec(T) = \mathbf{1}_n \tag{28}$$

where $\mathbf{1}_l$ is the l-dimensional column vector, I_l is the $l \times l$ identity matrix. We need to reshape the optimal $rvec(T)$ to obtain the matrix T for the final result.

Algorithm 2: AOC-KKmeans-L1/L2

Input: The dataset for clustering $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, kernel function K, aggregating matrix A, aggregated labels B;
Output: $c \times n$ 0-1 membership matrix T;
Initialize $T = (A^{\dagger}B)^T$;
repeat
  Calculate D using (11)
  switch *Loss function* L **do**
    case L_1
      Solve the linear programming (24)-(28) to upgrade T;
    end
    case L_2
      Solve the quadratic programming (29)-(31) to upgrade T;
      refine T;
    end
  end
until T *converges* ;

For AOC-KKmeans-L2, the sub-optimization problem when D is fixed can be written into the following matrix form

$$\min_{rvec(T)} \tfrac{1}{2}\lambda(rvec(T))^T(I_c \otimes (A^TA))(rvec(T)) + (rvec(D) - \lambda vec(A^TB))^T rvec(T) \quad (29)$$

$$0 \le rvec(T) \le 1 \quad (30)$$

$$(\mathbf{1}_c^T \otimes I_n)rvec(T) = \mathbf{1}_n \quad (31)$$

This is a standard formulation of quadratic optimization, which can also be solved efficiently. Then we reshape $rvec(T)$ to get T. We still use the continuous constraint (30) instead of the integer constraint. However, there is no justification for the equivalence as Theorem 1. Thus, by solving this quadratic optimization, we are making approximations. The final partition should be decided according to the maximum value in each row of T. We call this step as "refine T" in Alg. 2.

D. Transduction and Induction

The above proposed AOC-KKmeans algorithms are transductive, since they only aim to reveal the labels of the training samples, while cannot predict for the unseen samples, which is the goal of induction. We can train a supervised classifier for induction using the training samples and the revealed labels by AOC-KKmeans. There is an abundance of literature ([3]) for this task. The training for induction may be somehow connected with the previous AOC-KKmeans step. For example, if we choose to train an inductive classifier with kernel, it would be natural to use the same kernel as AOC-KKmeans.

V. Experiments, Discussion and Application

A. Overview

In Subsection V-B and V-C we conduct our experiments on two UCI datasets ([6]), namely *ionosphere* and *wine*. The first one is of binary classification case, and it is also used in [1]. The second one is of multi-class case which is not experimentally covered in [1]. The two datasets are originally collected for supervised learning, thus there are no natural bags generated specifically for the AOC problem. We need to create the aggregation ourselves. There are two parameters r and n_b we vary for the aggregation in our experiments. r is the "randomness" within bags ([1]) and n_b is the number of samples per bag.

Our algorithms include AOC-KKmeans-L1 and AOC-KKmeans-L2. We have also realized AOC-kNN and AOC-ANN for comparison. We did not compare with AOC-SVM mainly because it is too time-consuming. For the *ionosphere* dataset with 351 samples for training, the problem cannot be solved in one day.

We evaluate the performances of the four algorithms in both transductive and inductive settings. For transductive setting, we take all the samples for training, and we use the four algorithms to reveal the labels of them. The error rate is used as the criterion for evaluating the performances. For inductive setting, we randomly pick 80% of the dataset for training, and test on the remaining 20% to see the error rate. AOC-kNN and AOC-ANN can be both transductive and inductive. On the other hand, we need to train subsequent inductive classifiers for our two algorithms. We choose kNN and SVM using the same RBF kernel with AOC-KKmeans.

Each setting of our experiments is repeated 100 times to give average error rate and runtime. The differences between settings are due to different r, n_b as well as the randomly divided dataset for training and testing. Some discussion is made on the results of the experiments in Subsection V-D.

Finally in Subsection V-E, we propose a new setting for patch extraction in CBIR using AOC algorithm. We test it on images from LabelMe dataset ([7]) and show the results.

B. Ionosphere Dataset

Ionosphere is a dataset targeting free electrons in the ionosphere and it was collected in a radar system in Goose Bay, Labrador. There are altogether 351 samples in the dataset, for each of which there are 34 features. The labels of the samples can only be "Good" or "Bad", indicating whether that sample reveals the structure of the ionosphere or not. The experimental results of error rates and run time are reported in Fig. 1, 2 and Table II.

C. Wine Dataset

wine contains 178 instances of wine belonging to 3 different classes, each of which represents a wine cultivar. There are 13 features from chemical analysis for each sample. We show the results in Fig. 3, 4 and Table III.

Algorithm	Time/s
AOC-kNN	0.0266 ± 0.0064
AOC-ANN	18.7700 ± 1.7241
AOC-KKmeans-L1	0.2234 ± 0.0891
AOC-KKmeans-L2	0.3964 ± 0.1640

Table II
AVERAGE RUNTIME FOR THE FOUR ALGORITHMS ON *ionosphere* WITH THE TRANSDUCTION EXPERIMENTAL SETTING.

Algorithm	Time/s
AOC-kNN	0.0056 ± 0.0020
AOC-ANN	11.5055 ± 1.0855
AOC-KKmeans-L1	0.0818 ± 0.0144
AOC-KKmeans-L2	0.1796 ± 0.0355

Table III
AVERAGE RUNTIME FOR THE FOUR ALGORITHMS ON *wine* WITH THE TRANSDUCTION EXPERIMENTAL SETTING.

D. Discussion

From the results on *ionosphere* and *wine*, we can see that

1) Our two algorithms generally outperform AOC-kNN and AOC-ANN when evaluating with error rate.
2) Considering the runtime for the four algorithms, AOC-kNN is the fastest, with our two algorithms following close after. AOC-ANN is much more slower, which is inherited from the original BP network.
3) The performances of algorithms are degraded when the number of instance per bag n_b increases in general.
4) The increase in randomness r deteriorate the performances of the four algorithms (especially for AOC-kNN and AOC-ANN) in the transduction setting quite obviously, while has only slight effect in the induction setting.

E. Application: Patch Extraction for Content Based Image Retrieval

One important step of CBIR is to generate a codebook ([8]) for the specified content (e.g. trees, water). For this step, we need to extract a great amount of patches for that content from labeled training images. The most common way of labeling training images is to use polygon or other complicated boundary to enclose the specific content manually. However, this could be very time-consuming if we want to get a comprehensive collection. Here we suggest a possible novel way to deal with this problem via AOC. When given a training set, we only give an aggregated label for each image, namely to what ratio approximately the concerned content occupies in the whole image. This could be done very quickly according to the first impression. Then we randomly extract several patches from the images according to a uniform distribution as training samples, and deem the patches from the same image as one bag. By using the (transductive) AOC algorithm, we can give the labels for the patches. We conduct an experiment for extracting patches of trees. The training set contains 10 labeled images from LabelMe dataset ([7]). We extract 300 7×7 patches for each 256×256 image as training samples, and we use the raw features by expanding the patches into vectors. The original images and results by AOC-KKmeans-L1 are shown in Fig. 5. We use the RBF kernel with parameters tuned. One can see that the patches extracted correspond with the tree content well.

VI. CONCLUSION

In this paper, we set up a framework based on kernel K-means for the AOC problem, and propose two new algorithms. Our algorithms can be used for both binary and multi-class cases. We conduct experiments on two datasets whose results suggest the advantage of our methods. We also propose a new setting of patch extraction for CBIR with good experimental results shown.

ACKNOWLEDGMENT

This work is supported by NSFC (Grant No. 60675009).

REFERENCES

[1] D. Musicant, J. Christensen, and J. Olson, "Supervised learning by training on aggregate outputs," in *Seventh IEEE International Conference on Data Mining, 2007. ICDM 2007*, 2007, pp. 252–261.

[2] I. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM New York, NY, USA, 2004, pp. 551–556.

[3] R. Duda, P. Hart, and D. Stork, *Pattern classification*. Wiley New York, 2001.

[4] A. Dempster, N. Laird, D. Rubin *et al.*, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[5] B. Scholkopf and A. Smola, *Learning with kernels*. MIT press Cambridge, Mass, 2002.

[6] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[7] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "LabelMe: a database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1, pp. 157–173, 2008.

[8] L. Fei-Fei and P. Perona, "A Bayesian hierarchical model for learning natural scene categories," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, vol. 2, 2005.

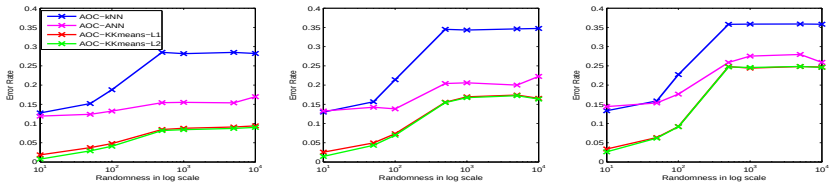

Figure 1. Results of experiments on *ionosphere*, with transduction setting. $n_b = 5, 10, 30$ respectively.

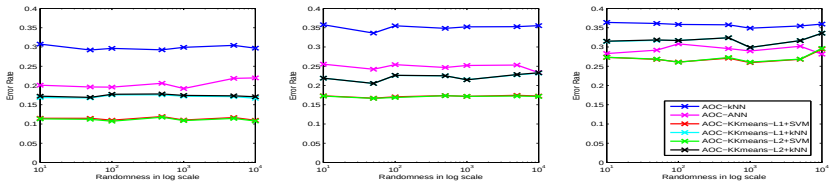

Figure 2. Results of experiments on *ionosphere*, with induction setting. $n_b = 5, 10, 30$ respectively.

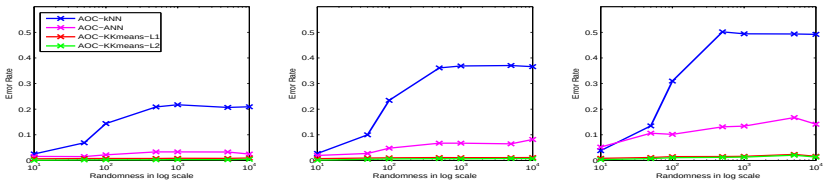

Figure 3. Results of experiments on *wine*, with transduction setting. $n_b = 5, 10, 20$ respectively.

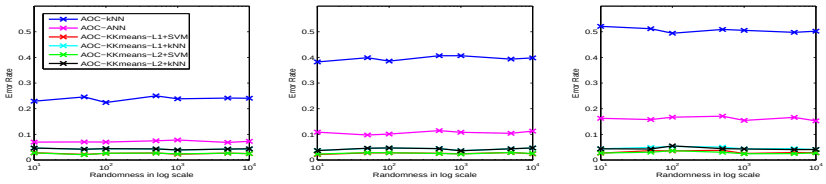

Figure 4. Results of experiments on *wine*, with induction setting. $n_b = 5, 10, 20$ respectively.

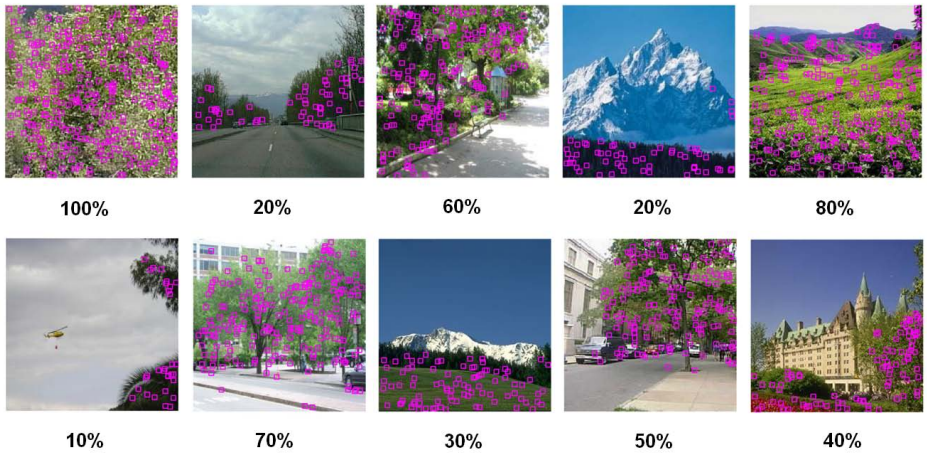

Figure 5. Application of AOC for tree patches extraction. We use 10 images containing trees as training set, with the approximate ratios of trees against the whole images labeled below. The extracted patches (marked by magenta frames) are given by AOC-KKmeans-L1.