Active Learning with the Probabilistic RBF Classifier

Constantinos Constantinopoulos* and Aristidis Likas

Department of Computer Science, University of Ioannina, GR 45110 Ioannina, Greece
ccostas@cs.uoi.gr, arly@cs.uoi.gr

Abstract. In this work we present an active learning methodology for training the probabilistic RBF (PRBF) network. It is a special case of the RBF network, and constitutes a generalization of the Gaussian mixture model. We propose an incremental method for semi-supervised learning based on the Expectation-Maximization (EM) algorithm. Then we present an active learning method that iteratively applies the semi-supervised method for learning the labeled and unlabeled observations concurrently, and then employs a suitable criterion to select an unlabeled observation and query its label. The proposed criterion selects points near the decision boundary, and facilitates the incremental semi-supervised learning that also exploits the decision boundary. The performance of the algorithm in experiments using well-known data sets is promising.

1 Introduction

Active learning a classifier constitutes a special learning problem, where the training data are actively collected during the training. The training data are available as a stream of classified observations, but the information they carry is controlled from the classifier. The classifier determines regions of interest in the data space, and asks for training data that lie in these regions. The importance of active learning is well established, see [1] for a study on the increase of classifier's accuracy as the number of labeled data increases. Various active learning methods have been suggested; in [2] a learning method for Gaussian mixture models [3] is proposed, that selects data that minimize the variance of the learner. In [4] active learning for a committee of classifiers is proposed, which selects data for which the committee members disagree. Based on this selection method, in [5] they propose the use of available unclassified data by employing EM [6] to form a better selection criterion, that is used to train a naive Bayes classifier. In [7] they train Gaussian random fields and harmonic functions, and select data based on the estimated expected classification error.

* This research was co-funded by the European Union in the framework of the program "Heraklitos" of the "Operational Program for Education and Initial Vocational Training" of the 3rd Community Support Framework of the Hellenic Ministry of Education, funded by 25% from national sources and by 75% from the European Social Fund (ESF).

S. Kollias et al. (Eds.): ICANN 2006, Part I, LNCS 4131, pp. 357–366, 2006.

Our work concentrates on a variation of the active learning scenario called the *pool-based* active learning, also studied in [5,7]. In this case a set of labeled and unlabeled observations is available right from the start. During training we are allowed to iteratively query the label of unlabeled points, and use the acquired labels to improve the classifier. In practice this scenario is important when querying a field expert is expensive, as in medical diagnosis, or when there is a huge quantity of unlabeled data that prohibits thorough labeling, as in text classification. The intuition behind pool-based learning is that the unlabeled data can be exploited to construct a more detailed generative model for the data set. Thus this problem is closely related to *semi-supervised* learning. Algorithms for semi-supervised learning have been proposed for Gaussian mixtures in [8,9], as well as for the RBF network [10]. So it has been established that unlabeled data can reveal useful information for the distribution of the labeled data.

We concentrate here on the pool-based active learning of the probabilistic RBF (PRBF) classifier [11,12]. It is a special case of RBF network [13] that computes at each output unit the density function of a class. It adopts a cluster interpretation of the basis functions, where each cluster can generate observations of any class. This is a generalization of a Gaussian mixture model [3,13], where each cluster generates observations of only one class. In [14] an incremental learning method based on EM for supervised learning is proposed. In this work we propose an incremental learning method based on EM for semi-supervised learning. We are facilitated by the fact that each node of the PRBF describes the local distribution of potentially all the classes. For the unlabeled data we can marginalize the class labels from the update equations of EM, to use both labeled and unlabeled data in parameter estimation.

In the following section we describe an incremental algorithm for the semi-supervised training of the PRBF based on EM. In section 3 we use this algorithm to tackle the problem of active learning. Next in section 4 we present the results from our experimental study. Some discussion in section 5 concludes this work.

2 Semi-supervised Learning

Assume a set of labeled observations $X = \{(x^n, y^n)|\ n = 1, \ldots, N\}$ and a set of unlabeled observations $X_\emptyset = \{x^n|\ n = 1, \ldots, N_\emptyset\}$. The labeled observations have an "input" part $x \in \Re^d$, and an "output" part $y \in \{1, \ldots, K\}$ in the case of a classification task with K classes. This "output" part (called label) assigns an observation to one class, and in the case of unlabeled observations is missing. Let Ω be the joint set of labeled and unlabeled observations, i.e. $\Omega = X \cup X_\emptyset$. Moreover we can separate X according to the "output" labels in K disjoint sets $X_k = \{(x^n, y^n)|\, y^n = k, n = 1, \ldots, N_k\}$ one for each class, then $\Omega = \bigcup_k X_k \cup X_\emptyset$.

Adopting the Bayes decision rule, a classifier assigns a new unlabeled observation $x^\star$ to the class $k^\star$ with maximum posterior probability. If we drop the part of the posterior that depends only on $x^\star$, then

$$k^\star = \arg\max_k p(x^\star|k)p(k) \tag{1}$$

where $p(x|k)$ is the class conditional distribution of observations from class k, and $p(k)$ is the prior probability of this class. For two classes k and k' there is a *decision boundary* $p(x|k)p(k) = p(x|k')p(k')$ that divides the space of the observations.

To describe class conditional distributions we employ the PRBF network. For input x the class conditional probability $p(x|k)$ is the k-th output of a PRBF with J basis functions

$$p(x|k) = \sum_{j=1}^{J} p(j|k)\, p(x|j) \tag{2}$$

The coefficients $p(j|k)$ are non-negative and $\sum_j p(j|k) = 1$, while each basis function is a Gaussian

$$p(x|j) = \frac{1}{(2\pi\sigma_j^2)^{d/2}} \exp\{-\frac{1}{2}(x-\mu_j)^T(x-\mu_j)/\sigma_j^2\} \tag{3}$$

with mean $\mu_j \in \Re^d$ and variance σ_j^2. In order to find estimates for the parameters of the network

$$\theta = \{p(k), p(j|k), \mu_j, \sigma_j |\, j = 1, \ldots, J,\ k = 1, \ldots, K\}$$

we maximize the joint likelihood, as in [10]. Assuming i.i.d. observations, the joint log-likelihood $\mathcal{L}$ of labeled and unlabeled data is

$$\begin{aligned}\mathcal{L} &= \log p(\Omega) = \log \prod_k \prod_{x \in X_k} p(x,k) \prod_{x \in X_\emptyset} p(x) \\ &= \sum_k \sum_{x \in X_k} \log p(k) \sum_j p(j|k)p(x|j) + \sum_{x \in X_\emptyset} \log \sum_k p(k) \sum_j p(j|k)p(x|j). \end{aligned} \tag{4}$$

For the maximization of $\mathcal{L}$ we use the Expectation-Maximization (EM) algorithm [6]. The EM is an iterative algorithm that is guaranteed to converge at a local maximum of the likelihood surface. It is employed in problems where *hidden variables* exist. These variables determine the solution of the problem, although are not observable. In our case the hidden variables define the node of the network that generated an observation, and the label of an unlabeled observation. In the following we formally derive the update equations of EM.

We introduce a hidden variable $z^{(x)}$ for each $x \in \Omega$ that assigns this observation to one class and one node of the network. Each $z^{(x)}$ is a binary $J \times K$ matrix, where $z_{jk}^{(x)} = 1$ if x is assigned to the k-th class and the j-th node. This assignment is unique, so that $\sum_j \sum_k z_{jk}^{(x)} = 1$. Moreover for a labeled observation (x,k) the corresponding $z^{(x)}$ is constrained so that $z_{j\ell}^{(x)} = 0$ for all (j,ℓ) with $\ell \neq k$. Thus a hidden variable can assign a labeled observation to any node but only one class. This does not hold for the case of unlabeled observations that can be assigned to any class and any node. Given the set of hidden variables $Z = \{z^{(x)} |\, \forall x \in \Omega\}$, we define the *complete* log-likelihood

$$\mathcal{Q} = \log p(\Omega, Z) = \log \prod_{x \in \Omega} \prod_k \prod_j [p(k)p(j|k)p(x|j)]^{z_{jk}^{(x)}} \tag{5}$$

Although we can not compute $\mathcal{Q}$ directly, as it depends on the unknown values of Z, we can compute its expectation $\langle\mathcal{Q}\rangle$ w.r.t. the distribution of Z. Since the expected value of $z_{jk}^{(x)}$ is equal to the joint posterior probability $p(j,k|x)$ that x is assigned to the j-th node and the k-th class, it follows that

$$\langle\mathcal{Q}\rangle = \sum_{x\in\Omega}\sum_{k}\sum_{j} p(j,k|x)\log\left\{p(k)p(j|k)p(x|j)\right\}. \tag{6}$$

The EM algorithm iterates two steps until convergence. During the *E-step* it computes the expectation of the complete log-likelihood $\langle\mathcal{Q}\rangle$, given the current estimate for the parameter vector θ. During the *M-step* it provides estimates θ that maximize $\langle\mathcal{Q}\rangle$. This procedure is guaranteed to converge at a local maximum of the joint log-likelihood $\mathcal{L}$.

Explicitly described, during the E-step we compute $p(j,k|x)$ for every $x\in\Omega$, $j\in\{1,\ldots,J\}$ and $k\in\{1,\ldots,K\}$ according to

$$p(j,k|x) = p(j|k,x)p(k|x). \tag{7}$$

If x is unlabeled then we compute $p(k|x)$ and $p(j|k,x)$ for every class k using Bayes theorem

$$p(k|x) = \frac{p(x|k)p(k)}{\sum_{\ell} p(x|\ell)p(\ell)} \tag{8}$$

$$p(j|k,x) = \frac{p(j|k)p(x|j)}{\sum_{i} p(i|k)p(x|i)}. \tag{9}$$

If x is labeled, then we exploit the information of the label and set

$$p(k|x) = \begin{cases} 1 \text{ if } x\in X_k \\ 0 \text{ if } x\notin X_k \end{cases} \tag{10}$$

and we compute $p(j|k,x)$ similarly

$$p(j|k,x) = \begin{cases} \frac{p(j|k)p(x|j)}{\sum_i p(i|k)p(x|i)} & \text{if } x\in X_k \\ 0 & \text{if } x\notin X_k \end{cases} \tag{11}$$

During the M-step we maximize $\langle\mathcal{Q}\rangle$ w.r.t. θ, given the current estimation of the joint posteriors. The solution for every $j\in\{1,\ldots,J\}$ and $k\in\{1,\ldots,K\}$ is

$$\mu_j = \frac{\sum_{x\in\Omega}\sum_k p(j,k|x)\, x}{\sum_{x\in\Omega}\sum_k p(j,k|x)} \tag{12}$$

$$\sigma_j^2 = \frac{1}{d}\frac{\sum_{x\in\Omega}\sum_k p(j,k|x)\,(x-\mu_j)^T(x-\mu_j)}{\sum_{x\in\Omega}\sum_k p(j,k|x)} \tag{13}$$

$$p(j|k) = \frac{\sum_{x\in\Omega} p(j,k|x)}{N_k + \sum_j \sum_{x\in X_\emptyset} p(j,k|x)} \tag{14}$$

$$p(k) = \frac{N_k + \sum_j \sum_{x\in X_\emptyset} p(j,k|x)}{N + N_\emptyset}. \tag{15}$$

An important aspect of network training is the estimation of the number of basis functions to be used. To tackle this we adopt the incremental approach proposed in [14] for supervised learning, that we modify suitably. It is an incremental method with two stages. We start with a network having only one node, whose parameters are easily estimated from the statistics of the training data. During the first stage we iteratively add new nodes to the network, until we reach the desired complexity. Then the second stage follows, where we split all the nodes in order to increase classification performance. In the next sections we give more details for the two stages.

2.1 Addition of Nodes

Given a network with M nodes we can construct a network with $M+1$ nodes. If the given class conditional density is $p(x|k)$, then adding a Gaussian node $q(x) = \mathcal{N}(x; \mu_q, \sigma_q^2)$ results in $\hat{p}(x|k)$ as follows

$$\hat{p}(x|k) = (1 - \alpha_k)\, p(x|k) + \alpha_k\, q(x) \tag{16}$$

where α_k is the prior probability that node q generates observations from class k. However we have to estimate α_k, the mean μ_q and variance σ_q^2 of q. Thus we search for parameters such that q is near the decision boundary. Good estimation of class conditional densities near the boundary is crucial for the performance of the classifier.

According to [14] we resort to a clustering method, namely the *kd-tree* [15]. The kd-tree is a binary tree that partitions a given data set. It is constructed recursively by partioning the data of each node in two subsets. Using only the labeled points, we initially partition the data in M subsets

$$X_j = \{(x,k)|\, (x,k) \in X,\, p(j|k,x) > p(i|k,x), \forall i \neq j\}$$

one for each node. Employing the kd-tree we repartition each of X_j in six subsets. These subsets result from the construction of a kd-tree with two levels. More levels would result in a lot of small clusters. We would like to avoid that, as we want to gradually shrink the size of the clusters. Moreover, in order to add the next node, we are going to employ the kd-tree again to partition all the data in smaller clusters. The statistics of the resulting subsets are probable estimates of μ_q and σ_q^2. The corresponding estimation of prior is $\alpha_k = p(j|k)/2$. Partitioning each node we create $6M$ sets of candidates $\theta_q = \{\alpha_k, \mu_q, \sigma_q^2\}$, so we have to select the most appropriate according to a criterion.

As proposed in [14], we compute the change of the log-likelihood $\Delta\mathcal{L}_k^{(q)}$ for class k after the addition of q

$$\begin{aligned} \Delta\mathcal{L}_k^{(q)} &= \frac{1}{N_k} \left(\log \hat{p}(x|k) - \log p(x|k)\right) \\ &= \frac{1}{N_k} \sum_{x \in X_k} \log \left\{ 1 - \alpha_k + \alpha_k \frac{q(x)}{p(x|k)} \right\}. \end{aligned} \tag{17}$$

We retain those θ_q that increase the log-likelihood of at least two classes and discard the rest. For each retained θ_q, we add the positive $\Delta\mathcal{L}_k^q$ terms to compute the total increase of the log-likelihood $\Delta\mathcal{L}_q$. The candidate $q^\star$ whose value $\Delta\mathcal{L}_{q^\star}$ is maximum consists the parameters of the node that will be added to the current network, if this maximum value is higher than a prespecified threshold. Otherwise, we consider that the attempt to add a new node is unsuccessful. We set this threshold equal to 0.01 after experimentation, in order to avoid the addition of nodes with negligible effects on the performance of the network. So we chose a small value, to also prevent the premature termination of the procedure.

After the successful addition of a new node we apply the semi-supervised EM, as described in the previous section. This procedure can be applied iteratively, in order to add the desired number of nodes to the given network. Figure 1 illustrates the addition of the first two nodes. The initial network with only one node is illustrated in Figure 1(a). The six candidate nodes and the chosen node are illustrated in Figure 1(b) and Figure 1(c) correspondingly. Figure 1(d) illustrates the network after the application of semi-supervised EM.

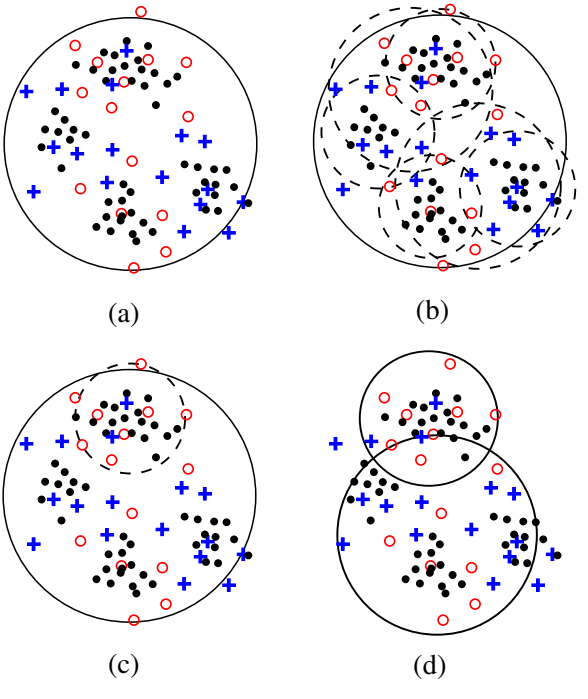

Fig. 1. Addition of the first two nodes. The nodes of the network are drawn with solid lines, and the candidate nodes with dashed lines. The dots represent the unlabeled observations in a two-class problem.

2.2 Splitting of Nodes

After the stage of adding nodes, there may be nodes of the network located to regions with overlapping among classes. In order to increase the generalization performance of the network we follow the approach suggested in [16], and split each node. During this stage we use both supervised and unsupervised observations. We evaluate the joint posterior probabilities $p(j,k|x)$ for a node, and define if it is responsible for observations of more than one class. If $\sum_{x\in\Omega} p(j,k|x) > 0$, then we remove it from the network, and add a separate node for the k-th class. So finally each node is responsible for only one class. Splitting a node $p(x|j)$, the resulting node for class k is a Gaussian $p(x|j,k)$ with mean μ_{kj}, variance σ^2_{kj} and mixing weight $p(j|k)$. These parameters are estimated according to:

$$\mu_{kj} = \frac{\sum_{x\in\Omega} p(j,k|x)\, x}{\sum_{x\in\Omega} p(j,k|x)} \tag{18}$$

$$\sigma^2_{kj} = \frac{1}{d}\frac{\sum_{x\in\Omega} p(j,k|x)\,(x-\mu_{kj})^T(x-\mu_{kj})}{\sum_{x\in\Omega} p(j,k|x)} \tag{19}$$

$$p(j|k) = \frac{\sum_{x\in\Omega} p(j,k|x)}{N_k + \sum_j \sum_{x\in X_\emptyset} p(j,k|x)}. \tag{20}$$

Consequently the class conditional density is estimated as

$$p(x|k) = \sum_j p(j|k)p(x|j,k). \tag{21}$$

In the case of a training set where all the points are labeled, the class conditional likelihood is increased for all classes after splitting as proved in [16]. However in the semi-supervised case we cannot guarantee that splitting increases the joint likelihood.

3 Active Learning

In the previous section we described an incremental algorithm for training a PRBF network using labeled and unlabeled observations. In the following we incorporate the algorithm in an active learning method, where we iteratively select an unlabeled point and query its label. After its label is given, we add the labeled point in the labeled set and train the network again. The crucial point is to pick a point that greatly benefits the training of our classifier. We propose the selection of a point that lies near the classification boundary. In this way we facilitate the iterative addition of basis functions on the classification boundary, as described in the previous section.

As a criterion of selecting a suitable point we propose the ratio of class posteriors. For each unlabeled observation $x \in X_\emptyset$ we compute the class posterior $p(k|x)$ for every class, and then find the two classes with the largest posterior values:

$$\kappa_1^{(x)} = \arg\max_k p(k|x),\ \kappa_2^{(x)} = \arg\max_{k\neq\kappa_1^{(x)}} p(k|x). \tag{22}$$

We choose to ask for the label of $\hat{x}$ that exhibits the smallest ratio of largest class posteriors:

$$\hat{x} = \arg\min_{x \in X_\emptyset} \log \frac{p(\kappa_1^{(x)}|x)}{p(\kappa_2^{(x)}|x)}. \tag{23}$$

In this way we pick the unlabeled observation that lies closer to the decision boundary of the current classifier. Note that according to (1) we classify observations to the class with the maximum class posterior. Thus for some x on the decision boundary holds that $p(\kappa_1^{(x)}|x) = p(\kappa_2^{(x)}|x)$. Consequently if an observation approaches the decision boundary between two classes, then the corresponding logarithmic ratio of class posteriors tends to zero.

Summarizing the presented methodology, we propose the following active learning algorithm:

1. Input: The set X of labeled observations, the set $X_\emptyset$ of unlabeled observations, and a degenerate network $PRBF_{J=1}$ with one basis function.
2. For $s = 0, \ldots, S-1$
   (a) Add one node to the network $PRBF_{J+s}$ to form $PRBF_{J+s+1}$.
   (b) Apply EM until convergence for semi-supervised training of $PRBF_{J+s+1}$.
3. For $s = 0, \ldots, S$
   (a) Split the nodes of $PRBF_{J+s}$ to form $PRBF_{J+s}^{split}$.
4. Select the network $PRBF_{J^\star}^{split} \in \{PRBF_J^{split}, \ldots, PRBF_{J+S}^{split}\}$ that maximizes the joint likelihood.
5. Set the current network: $PRBF_J = PRBF_{J^\star}$.
6. If $X_\emptyset$ is empty go to step 7, else
   (a) Pick an unlabeled observation $\hat{x}$ according to (23), and ask its label $\hat{y}$.
   (b) Update the sets: $X = X \cup \{(\hat{x}, \hat{y})\}$ and $X_\emptyset = X_\emptyset \setminus \{\hat{x}\}$.
   (c) Go to step 2.
7. Output: Split the nodes of $PRBF_J$ to form the output network $PRBF_J^{split}$.

In all our experiments we use $S = 1$, thus we try to add one node at each iteration of the active learning.

4 Experiments

For the experimental evaluation of our method we used three data sets, available from the UCI repository. The first is the "segmentation" set, that consists of 2310 points with 19 continuous features in 7 classes. The second is the "waveform" set, that consists of 5000 points with 21 continuous features in 3 classes. The last is the "optical digits" set, that consists of 5620 points with 62 continuous features in 10 classes. All the data sets were standardized, so that all their features exhibit zero mean and unit standard deviation. In all experiments we applied our algorithm starting with 50 uniformly selected labeled points. We treated the rest as a pool of unlabeled points, and we actively selected 400 more. Each experiment was repeated five times, and we computed the average generalization error on a separate test set that contained the 10% of the original data set. Figure 2

illustrates the average generalization error and the average number of PRBF nodes after each added label. The results are satisfactory, as the generalization error almost halved in all cases after the addition of 50 labels. After the addition of 300 labels the error had converged, and the addition of more labels offered little improvement. After the addition of 400 labels, the average error for the "segmentation" data set was 0.156, for the "waveform" data set was 0.091, and for the "optical digits" data set was 0.089. For comparison, we also applied the supervised learning method proposed in [14] using the original data sets. The generalization error of the resulting PRBF for the "segmentation" data set was 0.246, for the "waveform" data set was 0.142, and for the "optical digits" data set was 0.07. Concluding, we note that the number of nodes converged slower than the error, but eventually it also reached a plateau. The average number of nodes after the addition of 400 labels was 285.2 for the "segmentation" data set, 294.6 for the "waveform" data set, and 509 for the "optical digits" data set.

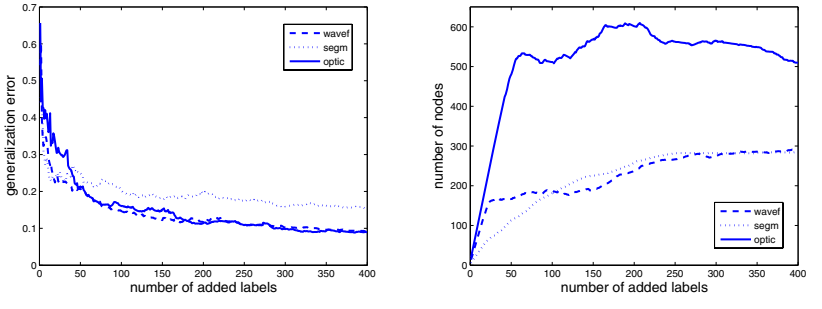

Fig. 2. The average generalization error (left), and the average number of network nodes (right) for pool-based active PRBF learning

5 Discussion

We have proposed an algorithm for the active learning of the PRBF classifier. We derived an EM algorithm for semi-supervised training of PRBF, and an incremental variation that sequentially adds nodes to the network. We use this method to estimate class conditional densities for pool-based active learning.

The experimental results are encouraging, and slight modifications of the method may further improve its performance. For example we could acquire the labels for a bunch of unlabeled observations, before we try to add a new node. The most important issue for consideration is the time complexity of the algorithm, e.g. it takes almost two hours to solve the "waveform" data set with a MATLAB implementation on a standard personal computer. A method to decrease the number of splits in each iteration would improve the execution time significantly. Another interesting issue concerns the complexity of the resulting network. We note that the interpretation of the network weights as probabilities alleviates the problem, as it forces many weights to near zero values and

overfitting is avoided. However we could use a validation set for better model selection.

Our future plans include a more detailed study of the method, and elaboration on several of our choices, with the most important being the comparison with other selection methods for the active acquisition of class information. Also we plan to consider the problem of new class discovery, as a similar task that we would like to tackle.

References

1. Castelli, V., Cover, T.: On the exponential value of labeled samples. Pattern Recognition Letters **16** (1995) 105–111
2. Cohn, D., Ghahramani, Z., Jordan, M.: Active learning with statistical models. Journal of Artificial Intelligence Research **4** (1996) 129–145
3. McLachlan, G., Peel, D.: Finite Mixture Models. John Wiley & Sons (2000)
4. Freund, Y., Seung, H.S., Shamir, E., Tishby, N.: Selective sampling using the query by committee algorithm. Machine Learning **28** (1997) 133–168
5. McCallum, A.K., Nigam, K.: Employing EM in pool-based active learning for text classification. In Shavlik, J.W., ed.: Proc. 15th International Conference on Machine Learning, Morgan Kaufmann (1998)
6. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood estimation from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B **39**(1) (1977) 1–38
7. Zhu, X., Lafferty, J., Ghahramani, Z.: Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions. In: Proc. 20th International Conference on Machine Learning. (2003)
8. Ghahramani, Z., Jordan, M.: Supervised learning from incomplete data via an EM approach. In Cowan, J.D., Tesauro, G., Alspector, J., eds.: Advances in Neural Information Processing Systems 6, Morgan Kaufmann (1994)
9. Tadjudin, S., Landgrebe, A.: Robust parameter estimation for mixture model. IEEE Trans. Geoscience and Remote Sensing **38** (2000) 439–445
10. Miller, D., Uyar, H.: Combined learning and use for a mixture model equivalent to the RBF classifier. Neural Computation **10** (1998) 281–293
11. Titsias, M.K., Likas, A.: Shared kernel models for class conditional density estimation. IEEE Trans. Neural Networks **12**(5) (2001) 987–997
12. Titsias, M.K., Likas, A.: Class conditional density estimation using mixtures with constrained component sharing. IEEE Trans. Pattern Anal. and Machine Intell. **25**(7) (2003) 924–928
13. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press (1995)
14. Constantinopoulos, C., Likas, A.: An incremental training method for the probabilistic RBF network. IEEE Trans. Neural Networks **to appear** (2006)
15. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM **18**(9) (1975) 509–517
16. Titsias, M.K., Likas, A.: Mixture of experts classification using a hierarchical mixture model. Neural Computation **14**(9) (2002) 2221–2244