SVM-KM: speeding SVMs learning with *a priori* cluster selection and k-means

Marcelo Barros de Almeida, Antônio de Pádua Braga
Universidade Federal de Minas Gerais
Department of Electronics Engineering
{barros,apbraga}@cpdee.ufmg.br

João Pedro Braga
Universidade Federal de Minas Gerais
Department of Chemistry
jpbraga@oxigenio.qui.ufmg.br

Abstract

A procedure called SVM-KM, based on clustering by k-means and to accelerate the training of Support Vector Machine, is the main objective of the present work. During the Support Vector Machines (SVMs) optimization phase, training vectors near the separation margins, are likely to become support vector and must be preserved. Conversely, training vectors far from the margins are not in general taken into account for SVM's design process. SVM-KM groups the training vector in many cluster. Clusters formed only by a vector that belongs to the same class label can be disregard and only cluster centers are used. On the other hand, clusters with more than one class label are unchanged and all training vectors belonging to them are considered. Cluster with mixed composition are likely to happen near the separation margins and they may hold some support vectors. Consequently, the number of vectors in a SVM training is smaller and the training time can be decreased without compromising the generalization capability of the SVM.

1 Introduction

Support Vector Machines (SVMs) have became an useful tool in pattern recognition problems, with application in many areas like SPAM categorization [5], Texture Recognition [2], Gene Expression Analysis [4], Text Categorization [9] and Hand-written Digit Recognition [18].

The SVM design consists of solving a quadratic program (QP) with linear constraints which depends only on the training vectors and on few kernel parameters. The QP solution provides necessary information to choose, among all data, the most important vectors which will define the separation hyper-plane between the classes, known as support vectors (SV).

In classification problems the support vectors represents only a small subset of the training vectors. Thus, the number of vectors used in the training phase will be decreased when unnecessary vectors are eliminated. This can be done by applying the k-means algorithm [12] to organize the training vector in several small clusters. Clusters away from the separation hyper-plane are exchanged by theirs centers whereas clusters near to this plane are unaffected. Consequently, the number of training vectors is decreased, speeding up the SVMs training and requiring less computational efforts without compromising the generalization capability of the SVM.

In Section 2 it is presented the formulation of the SVMs training problem and the Karush-Kuhn-Tucker conditions are discussed. Section 3 is about some strategies to solve the QP problem arising from SVMs and how SVM-KM can be used in this process. Computational experiments are seen in Section 5 and the results are discussed in Section 6.

2 SVMs formulation

The main task of the SVMs training is to find an optimal hyper-plane that can separate the two class labels, represented as (-1) and $(+1)$:

$$\mathbf{w}^T\varphi(\mathbf{x}) + b = 0 \tag{1}$$

with minor classification errors between them.

The function $\varphi(\cdot)$ maps a vector $\mathbf{x}$ into a higher-dimensional space where they can be separated. The mapped vector are weighted by $\mathbf{w}$ and added to a bias term b to provide the SVMs output for a given input pattern. This hyper-plane can be found by a convex optimization problem as shown by [18]:

Given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, find the optimal values for the weight vector $\mathbf{w}$ and bias b such as the following constraints

$$y_i\mathbf{w}^T\varphi(\mathbf{x}_i) \geq 1 - \xi_i \quad \text{for} \quad i = 1, 2, \ldots, N \\ \text{and} \quad \xi_i \geq 0\,\forall\, i \tag{2}$$

1522-4899/00 $10.00 © 2000 IEEE

are satisfied, whereas the cost functional

$$\Phi(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N} \xi_i \qquad (3)$$

is minimized.

Slack variables ξ_i are introduced to allow some misclassification and C is a positive parameter provided by the user that controls the amount of misclassification.

Dual form [13, 3] of the above problem can be represented as (see [18] for a detailed deduction):

Given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, find the optimal Lagrange multipliers α_i for the problem:
Maximize

$$W(\boldsymbol{\alpha}) = \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{H}\boldsymbol{\alpha} \qquad (4)$$

subject to:

1. $\boldsymbol{\alpha}^T\mathbf{y} = 0$

2. $0 \leq \alpha_i \leq C$

where $H_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) = y_i y_j \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$ and $\boldsymbol{\alpha} = [\alpha_1 \ \alpha_2 \ \ldots \ \alpha_N]^T$. The kernel function is represented by $K(\cdot, \cdot)$.

Each training vector has associated to it a Lagrange multiplier, calculated during the training procedure. Depending on the Lagrange multiplier values, the vector is classified as support vector or not. The Karush-Kuhn-Tucker (KKT) for SVMs will doing classification according to the following rules ($f(\mathbf{x})$ is the SVM output for vector $\mathbf{x}$):

(i)	$\alpha_i = 0$	$y_i f(\mathbf{x}_i) > 1$	Ordinary vector
(ii)	$0 < \alpha_i < C$	$y_i f(\mathbf{x}_i) = 1$	SV on the margin
(iii)	$\alpha_i = C$	$y_i f(\mathbf{x}_i) < 1$	SV between margins

The Figure 1 shows this three conditions. Whenever the vector associated with null Lagrange multipliers are known *a priori*, they can be eliminated, not affecting the optimal hyper-plane and therefore reducing the training complexity.

3 Training SVMs

The QP problem solution is straightforward for small training sets is (typically less then a few hundred vectors) using standard QP packages, like MINOS, CPLEX, LOQO and the QP routine present in Matlab toolbox [16].

When the training data set used is too large, the memory space required to store the kernel matrix ($\mathbf{H}$) becomes large and therefore may not fit in this memory. Consequently, it

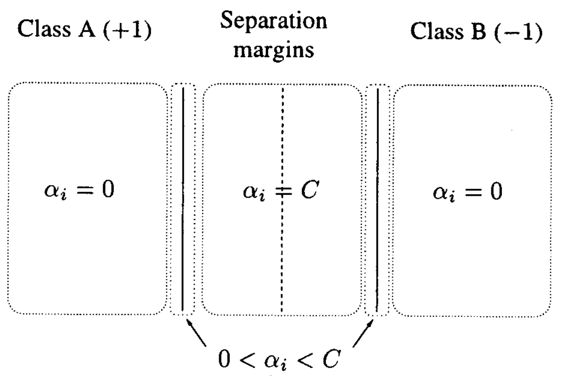

Figure 1. Correctly classified vectors are related to null Lagrange multipliers and they are placed out of the region between margins. These vectors are irrelevant in the future use of the SVMs. On the other hand, correctly classified vectors placed on the margins are associated with Lagrange multipliers in the range ($0 < \alpha_i < C$). Vectors between margins are related to Lagrange multipliers at the upper bound (C). These last two groups of vectors are known as support vectors and they will define the separation hyper-plane between the classes.

is necessary to use iterative methods or methods based on the decomposition of the QP problem.

Kernel Adatron algorithm [6] is a good example of iterative method which implements a gradient ascent strategy and optimizes one Lagrange multiplier in each turn. Methods that use only subsets of the QP problem are based on the idea of active sets [13] and tries to use heuristics to determine good subsets to optimize them. The chunk method [17], SVM Light [8], Successive Over Relaxation (SOR) [14] and Nearest Point Algorithm [11] are examples of these methods.

The Sequential Minimal Optimization (SMO) [15], an iterative process where two Lagrange multipliers are update at each iteration, allows an analytical solution for the two Lagrange multipliers. Heuristics are used to choose the two Lagrange multipliers. Furthermore SMO does not need to store the kernel Matrix in the memory. A C++ version of Platt's SMO [15] was implemented for the simulation that follows. This version is called SMOBR and the code can be downloaded from [1].

4 SVMs and k-means

Since the support vectors represents only a small subset of the training vectors, the k-means algorithm can be applied to clusters the vectors less important before the SVMs training. On the contrary, vectors near the separation margin must be preserved since they will define the optimal separation hyper-plane.

Therefore, SVMs training can be accelerated if it is applied the suggested procedure, called SVM-KM, described below:

1. Define a number k of clusters to be detected by k-means. This number will depend on the sparseness of the data and the available number of training vectors (N). For instance, if it is desired to group the vectors in clusters of five examples, the k value will be $N/5$.
2. Run k-means on the entire data set to detect the k cluster centers. Since the used k-means is not a supervised algorithm, it is necessary to verify if in the clusters found by k-means there exist more than one class label. Clusters formed only by vectors that belong to the same class label can be disregard and used only their centers. On the other hand, clusters with more than one class label are unchanged and all training vector belonging to them are considered.
3. With the new training data set defined, SVMs training can be carried on.

The version of k-means implemented is suggested by Lloyd [12]. It starts with a predefined number of cluster whose centers are defined by assigning training vector randomly. For each training vector the nearest center is calculated. Next, the new centers are computed using the mean of nearest vectors. This is performed until no more changes in average values are detected. The k-means used was written in C++ and can be obtained from [1].

5 Simulations

Two experiments using two-dimensional data, not linearly separable, where performed. Each experiment was repeated 100 times and an average of the most important parameters was taken. Simulations were carried on a Pentium 400MHz with 128MB of memory and running Linux. The experiments are described below.

5.1 First experiment

For the first experiment it was used a training set with 250 vectors with dimension 2 in which 100 vectors belonging to class (-1) and 150 vectors to class $(+1)$. The (-1) class has its vectors distributed around the centers (1,1) and (5,5), with standard deviation 0.9 and according to a Gaussian distribution. The class $(+1)$ has its centers around (1,5), (5,1) and (3,3), with the same standard deviation and distribution used in class (-1). The kernel chosen was RBF (Radial Basis Function [18]), given by

$$K(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{|\mathbf{x}-\mathbf{y}|_2^2}{2\sigma^2}\right) \qquad (5)$$

with $\sigma = 2$ and the value for the upper limit was $C = 5$. The k-means was run to find 50 centers (5 vectors per cluster). In the generalization test, 10000 vectors were used, according to the same distributions defined before.

Table 1 summarizes the average results for this experiment. Even considering the k-means time (0.073) the total time spent (0.71) is considerable smaller than SMO's time when using the complete training set. Besides this, the average of number support vectors was decreased. This will permit a faster evaluation of the separation hyper-plane in the future use of SVMs. Norm of $\mathbf{w}$ had also a small de-

Table 1. Average result for the experiment 1. TV and SV stand for training vectors and support vectors, respectively. MSE represents the mean-square error for generalization test.

	SMO	SVM-KM (K-MEANS+SMO)
CPU time (s)	2.24	0.71 (0.073+0.64)
$\|\mathbf{w}\|^2$	46.74	35.95
MSE	3.23	2.88
Num. of TV	250	94.72
Num. of SV	46.61	41.17
Success (%)	93.98%	93.72%

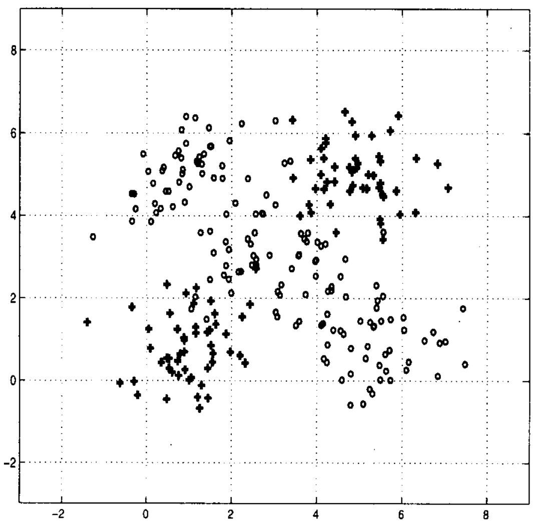

Figure 2. Complete training set for experiment 1

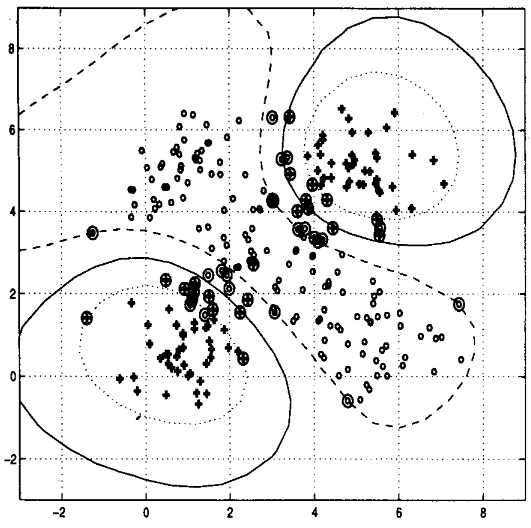

Figure 3. Non-linear decision boundaries in the input space for experiment 1.

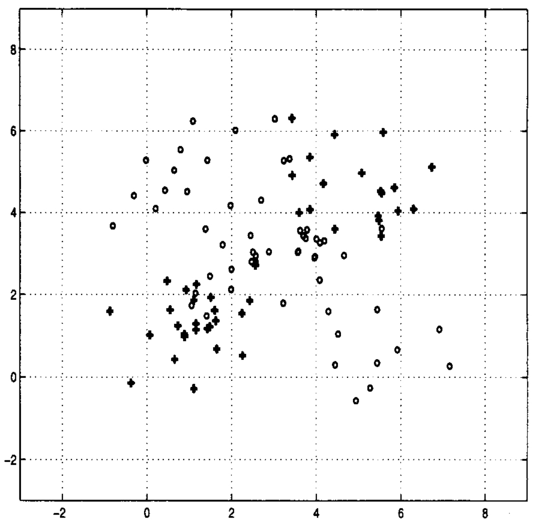

Figure 4. Reduced training set for experiment 1 with prior application of k-means

crease than the one obtained with the complete training set. The generalization capability of SVMs was decreased by 0.27%, whereas training time was speeded up by a factor of 3.13 and training set size was decreased by a factor of 2.73. Although the generalization capability has decreased the overall performance of the SVMs training was increased considerable.

Figures 2, 3, 4 and 5 illustrate the data sets and separation hyper-plane for this experiment. Note that the vectors near the separation margins are preserved and many support vectors are coincident (support vectors are marked with a circle around training vectors). The margins for SVM-KM are slightly smoother, resulting on less over-fitting to the data. In Figures 3 and 5, full contour is for hyper-plane equation (Equation 1) equals to 0, dashed contour for hyper-plane equation equals to $+1$ and dotted contour for hyper-plane equation equals to -1.

5.2 Second experiment

Another artificial data set was generated for this experiment which is similar to the one described in [10]. The training set consists of 1000 vectors distributed in a 4×4 checkerboard (see Figure 6) and 10000 test vectors. All vectors are drawn from an uniform distribution in the $[0,2] \times [0,2]$ range. The kernel chosen was RBF with $\sigma = 0.25$ and the value for the upper limit was $C = 20$. The k-means algorithm was run to find 100 centers (10 vectors per cluster).

Table 2 summarizes the average results for this experiment and the conclusions are similar to experiment 1. SVM-KM reduces the total training time with small decrease in the generalization capability of SVM. Furthermore the number of support vectors is smaller than the number of support vector obtained when training is performed with complete training set. Figures 6, 7, 8 and 9 show the data sets and separation hyper-planes for this experiment.

Table 2. Average result for the experiment 2. Labels as in Table 1

	SMO	SVM-KM (K-MEANS+SMO)
CPU time (s)	32.21	13.89 (0.89+13.00)
$\|\mathbf{w}\|^2$	1562.2	1349.9
MSE	1.78	1.99
Num. of TV	1000	497.15
Num. of SV	172.19	159.69
Success (%)	95.96%	94.92%

6 Discussion

Although k-means is an exhaustive technique for clustering, its joint application with SMO in SVMs training is feasible since the time spent is small when compared with the SMO training time using the complete training set. The k-means efficiency may decrease if the dimensionality of the vectors is large and a good relationship must be achieved.

Other important aspect is related to the generalization quality of the SVM trained with SVM-KM. In the tests per-

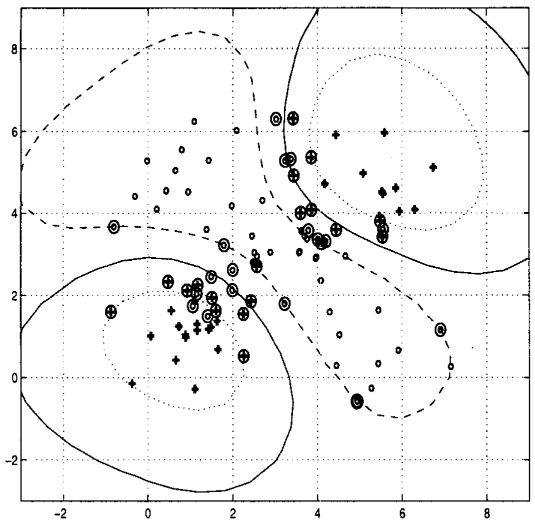

Figure 5. Non-linear decision boundaries in the input space for experiment 1 with prior application of k-means.

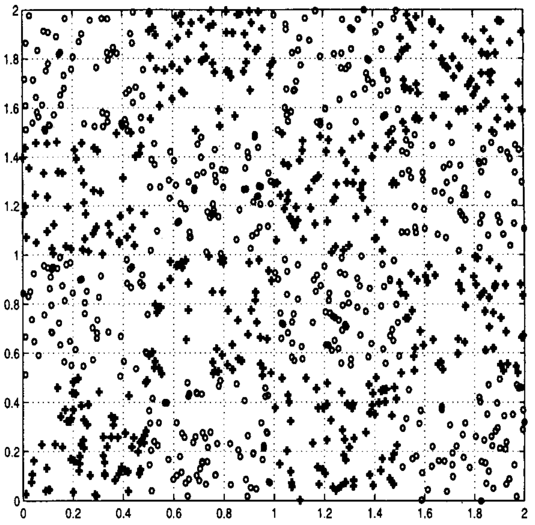

Figure 6. Complete training set for experiment 2

formed the kernel parameters were the same to the SVM training with and without SVM-KM. However, it would be better if new kernel parameters were specified to SVM-KM, since the support vector sets are slightly different. In this way better generalization rates would be obtained.

The spatial distribution of the training set may influence SVM-KM performance. Dense training sets are more suitable to applying SVM-KM than sparse ones. Moreover, the number of cluster parameters can be changed significantly depending on the training set density. However, this parameters are easy to be specified.

It should be realized that a noisy vector (misclassified) may force the selection of an entire cluster even if it is away from margins. In this case the training set after SVM-KM will be larger but this noisy vector can be eliminated by the posterior SVM training.

7 Conclusion

This work showed how k-means can be used to accelerate Support Vector Machines training. The main idea is to cluster vectors away from the separation margins and preserve vectors near it. With this procedure, the number of vectors in SVM training is smaller and the training time can be decreased without compromising the generalization capability of the SVM. Besides this, two simulations were performed to exemplify the proposed technique.

SVM-KM will be efficient when dealing with low dimensional and dense training sets. The training set size an be reduced and therefore SMO's training time decreased. Performance may be affected whenever dimension becomes larger since several distance evaluations are performed during k-means execution.

Acknowledgements

The authors would like to thank the Brazilian National Research Council (CNPq) and FAPEMIG for the financial support.

References

[1] M. B. d. Almeida. `http://litc.cpdee.ufmg.br/~barros/svm/smobr/`, 2000.

[2] O. Barzilay and V. Brailovsky. On domain knowledge and feature selection using a support vector machine. *Pattern Recognition Letters*, 20:475–484, 1999.

[3] M. Bazaraa and S. CM. *Nonlinear programming*. John Wiley and Sons, Inc, USA, 1979.

[4] M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of The National Academy of Sciences of The United States of America*, 97:262–267, 2000.

[5] H. Drucker, D. Wu, and V. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10:1048–1054, 1999.

[6] T.-T. Frieß, N. Cristianini, and C. Campbell. The kernel adatron algorithm: A fast and simple learning procedure for

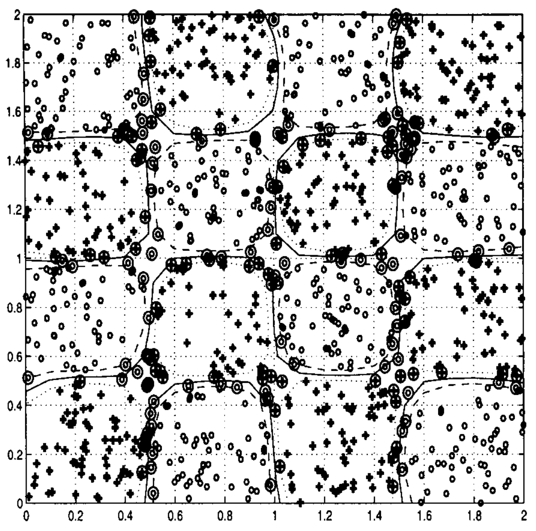

Figure 7. Non-linear decision boundaries in the input space for experiment 2.

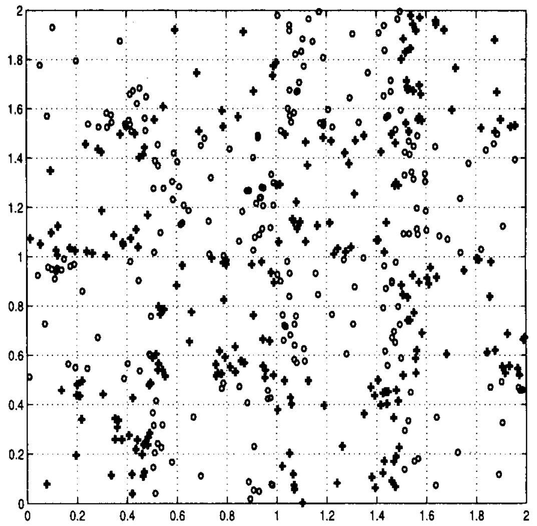

Figure 8. Reduced training set for experiment 2 with prior application of k-means

support vector machines. In *15th Intl. Conf. Machine Learning*. Morgan Kaufmann Publishers, 1998.

[7] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, Inc., 2nd edition, 1999.

[8] T. Joachims. *Advances in kernel methods: support vector machines*, chapter 11. MIT Press, Cambridge, MA, 1998.

[9] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137 – 142, Berlin, 1998. Springer.

[10] L. Kaufmann. *Solving the Quadratic Programming problem arising in Support Vector classification*, pages 147–168. MIT Press, Cambridge, MA, 1999.

[11] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. Technical report, Indian Institute if Science, Dept. of Computer Science and Automation, Bangalore, India, 1999.

[12] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:128–137, 1982.

[13] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, USA, 1986.

[14] O. Mangasarian and D. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10:1032–1037, 1999.

[15] J. C. Platt. *Advances in kernel methods: support vector machines*, chapter 12. MIT Press, Cambridge, MA, 1998.

[16] A. Smola and B. Scholkopf. `http://www.kernel-machines.org/`, 2000.

[17] V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer Verlag, New York, 1982.

[18] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons;, 1998.

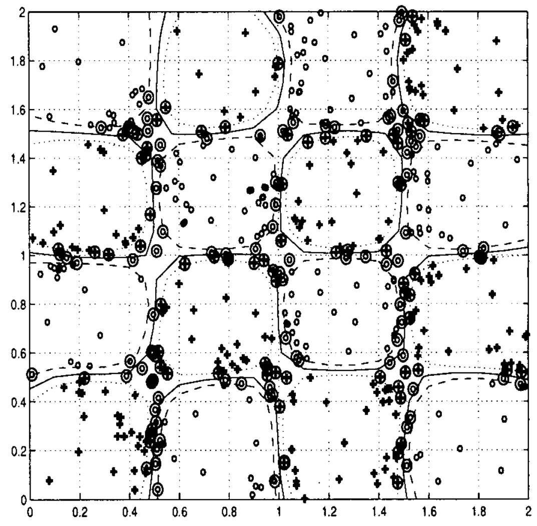

Figure 9. Non-linear decision boundaries in the input space for experiment 2 with prior application of k-means.