A Trainable Classifier via k Nearest Neighbors

I. Mora-Jiménez, A. Lyhyaoui, J. Arenas-García
A. Navia-Vázquez and A.R. Figueiras-Vidal
Dept. of Signal Theory and Communications
Univ. Carlos III de Madrid, Leganés-Madrid, Spain

Abstract. This paper introduces a new classifier derived from a variant of the k-Nearest Neighbor (kNN) rule. This classification scheme, which we call kNN Learning Vector Classifier (kNN-LVC), has a similar architecture to that of Learning Vector Quantizers (LVQs). In fact, both methods place in the observation space a set of centroids or prototypes with a limited area of influence; however, our approach finds optimal prototypes by optimizing a new discriminant function that considers the k nearest prototypes to a sample. Among kNN-LVC characteristics are its localized nature, easy training and interpretation, small storage requirements, and a very competitive performance. The proposed technique is benchmarked against other classifiers as kNN and LVQ. Experiments show good generalization capabilities and efficacy of our approach on datasets with enough number of data in relation to dimensionality.

Keywords. k Nearest Neighbors (kNN), Bayes Decision Rule, Prototype-based Classifier, Iterative Optimization.

1 Introduction

A desirable property of any classifier, apart from being a reliable system, is that it provides an understanding of its operation. The k Nearest Neighbor (kNN) rule [1], [4], [5] fulfills this condition. It is a simple, understandable and very intuitive method for classification: the class membership of a sample is chosen to be the majority class among the k most similar patterns to the sample. Besides, this classifier presents an asymptotic error rate less than twice the Bayes error probability [3], what is quite interesting if we consider the kNN rule does not assume any statistical model for the underlying problem.

The main drawback of the kNN rule is that, to classify a sample, it needs to calculate the similarity between that sample and all the training patterns. Therefore, this classifier involves a heavy computational cost, apart from the need of storaging all training samples. In fact, these are the reasons why its use is not very extended, even though it provides very good recognition rates.

The Learning Vector Quantization (LVQ) scheme [7] proposed by Kohonen can be seen as an alternative to reduce the amount of stored patterns while controlling the error. LVQ is a supervised classification technique that partitions data space into regions, each one defined by a labelled prototype or centroid. During the learning phase, prototypes' locations are tuned in order to represent the probability density function (pdf) of the data. After training, LVQ classifies a sample with the label of its nearest prototype.

Our work is directed towards building a classifier having the good characteristics from both kNN and LVQ methods, whereas achieving better performance. To do so, we first start from a modified kNN scheme, which takes into account not only the class of the k nearest patterns but also a measure of their closeness. Then, we derive a discriminant function and apply it to a pool of prototypes, obtaining a trainable version of the modified kNN rule. Optimization of this discriminant leads to the proposed k Nearest Neighbor Learning Vector Classifier (kNN-LVC). The main difference between LVQ and our scheme is that kNN-LVC does not pursue a model for the probability density function. Besides, the proposed classifier takes into account the closer prototype and the following runner-ups, what gives more potential to the classifier.

The outline of the paper is as follows. We first review the basics of the kNN rule and present our modified version. In Section 3 we derive kNN-LVC, together with a gradient based training scheme. Section 4 presents experimental results using several benchmarking problems and compare the performance of kNN-LVC with competitive classifiers as classical kNN and LVQs. Finally, we discuss the results and suggest further lines of work.

2 Classification via k Nearest Neighbors

Fundamentals of kNN rule [1], [4], [5] for classification come from the statistical approach to pattern recognition, where the purpose is to minimize the classification error.

One method for designing classifiers with the lowest error probability is the well-known Bayes' decision rule for minimum error [4]. Assuming $\mathbf{x}$ is a pattern in an n-dimensional feature space, this rule assigns class C_i (i: class index) to $\mathbf{x}$ if

$$P(C_i|\mathbf{x}) > P(C_j|\mathbf{x}) \quad \forall j \neq i \tag{1}$$

where the left-hand side of eq.(1) is the a posteriori probability of vector $\mathbf{x}$ to belong to class C_i. Using Bayes theorem, $P(C_i|\mathbf{x})$ can be expressed as

$$P(C_i|\mathbf{x}) = \frac{P(C_i)p(\mathbf{x}|C_i)}{p(\mathbf{x})} \tag{2}$$

where $P(C_i)$ denotes the prior probability of class C_i, $p(\mathbf{x}|C_i)$ is the pdf of data belonging to C_i, and $p(\mathbf{x})$ corresponds to the unconditional data pdf.

Introducing eq.(2) into (1), and for a two-class case ($i = 0, 1$), we have the following classification rule: if

$$P(C_1)p(\mathbf{x}|C_1) > P(C_0)p(\mathbf{x}|C_0) \tag{3}$$

then $\mathbf{x}$ is assigned to class C_1; otherwise class C_0 is selected. From now on, we will express this as

$$P(C_1)p(\mathbf{x}|C_1) \underset{C_0}{\overset{C_1}{\gtrless}} P(C_0)p(\mathbf{x}|C_0) \tag{4}$$

When dealing with a finite set of patterns with an unknown pdf, values in eq.(4) have to be substituted for their estimates. Assuming there are N_i samples for class C_i and N in total ($N = N_0 + N_1$), we can estimate the prior probabilities from the relative frequency

$$\hat{P}(C_i) = \frac{N_i}{N} \quad i = 0, 1 \tag{5}$$

As for the class conditional pdf, we propose to use the k Nearest Neighbor density estimator [1]. Let $d(\mathbf{x}, \mathbf{y})$ be the distance between $\mathbf{x}$ and $\mathbf{y}$ according to a certain

metric d; the kNN density estimation scheme first finds the k nearest neighbors to $\mathbf{x}$ according to metric d, and then determines the minimum volume $v(\mathbf{x})$ centered in $\mathbf{x}$ and containing these k samples. This way, the estimate is given by

$$\hat{p}(\mathbf{x}) \simeq \frac{k}{N v(\mathbf{x})} \tag{6}$$

Using eq.(6) to approximate the class conditional density [1], we have

$$\hat{p}(\mathbf{x}|C_i) = \frac{k_i}{N_i v(\mathbf{x})} \quad i = 0, 1 \tag{7}$$

where k_i is the number of samples from C_i among the k nearest neighbors ($k = k_0 + k_1$). This formulation leads to the classical kNN rule, which assigns vector $\mathbf{x}$ to the class with a higher k_i value. Optimal selection of parameter k depends on dataset size and dimensionality [1],[4],[5].

Fukunaga [5] proposes to proceed separately for each class, taking the k_1 nearest neighbors belonging to class C_1 and the corresponding k_0 from C_0 (with fixed values for k_0 and k_1), and then estimating the class-conditional densities as

$$\hat{p}(\mathbf{x}|C_i) = \frac{k_i}{N_i v_{k_i}(\mathbf{x})} \quad i = 0, 1 \tag{8}$$

where $v_{k_i}(\mathbf{x})$ is the volume centered in $\mathbf{x}$ encompassing the k_i selected samples of class C_i.

Independent selection of k_0 and k_1 yields some advantage over classical kNN rule, although a fixed selection for the whole space can result in some disadvantage for certain regions. On the other hand, it should be noted that this flexibility implies additional computational cost, since it is necessary to explore values for two parameters instead of one.

To avoid this extra complexity while preserving the advantages inherent to Fukunaga's scheme, we propose to use an analogous procedure: fixing k, the total number of samples to consider, and extracting values k_i from it. This way, values k_1 and k_0 will vary depending on the sample to classify.

Now, from equations (4), (5) and (8), it is immediate to get the following classification rule

$$\frac{k_1}{v_{k_1}(\mathbf{x})} \underset{C_0}{\overset{C_1}{\gtrless}} \frac{k_0}{v_{k_0}(\mathbf{x})} \tag{9}$$

If we accept to use hyperellipsoidal volumes and, for simplicity, the same sample covariance matrix for both classes, eq.(9) can be expressed as

$$\frac{k_1}{d_{k_1}^n} \underset{C_0}{\overset{C_1}{\gtrless}} \frac{k_0}{d_{k_0}^n} \tag{10}$$

Assuming Euclidean distance (represented by $||.||_2$) and spherical symmetry, we have the following version for the modified kNN rule

$$\frac{k_1}{||\mathbf{x} - \mathbf{x}^{(k_1)}||_2^n} \underset{C_0}{\overset{C_1}{\gtrless}} \frac{k_0}{||\mathbf{x} - \mathbf{x}^{(k_0)}||_2^n} \tag{11}$$

where $\mathbf{x}^{(k_i)}$ is the furthest neighbor belonging to class C_i and n is the dimensionality of the data.

The main property of our kNN version is that it takes advantage of the information about the distance to the k_i-th nearest sample of C_i, but it needs just one parameter to build the classifier. Nevertheless, it still demands large memory, as well as a substantial computational cost. To overcome these difficulties we devise the kNN Learning Vector Classifier, which is presented in the next section.

3 The proposed kNN Learning Vector Classifier (kNN-LVC)

After slight changes, the kNN version proposed in Section 2 is appropiate to design local classifiers with low storage requirements, in addition to allow gradient trainable schemes. It suffices to choose a pool of centroids $\{\mathbf{c}_0\}$ and $\{\mathbf{c}_1\}$ for classes C_0 and C_1, respectively, and introduce them in eq.(11). After some rearrangement we obtain

$$\frac{||\mathbf{x}-\mathbf{c}_0{}^{(k_0)}||_2^2}{k_0^{2/n}} - \frac{||\mathbf{x}-\mathbf{c}_1{}^{(k_1)}||_2^2}{k_1^{2/n}} \underset{C_0}{\overset{C_1}{\gtrless}} 0 \tag{12}$$

Assuming t is the target value for sample $\mathbf{x}$ ($t=1$ if $\mathbf{x}\in C_1$, $t=-1$ if $\mathbf{x}\in C_0$), we derive from (12) the following discriminant function D

$$D = t\left(\frac{||\mathbf{x}-\mathbf{c}_0{}^{(k_0)}||_2^2}{k_0^{2/n}} - \frac{||\mathbf{x}-\mathbf{c}_1{}^{(k_1)}||_2^2}{k_1^{2/n}}\right) \tag{13}$$

It is immediate to check that a sample $\mathbf{x}$ is correctly classified by this method if $D>0$. Therefore, our learning algorithm will modify progressively the centroids' positions to find a maximum of D. To do so, we apply the method of steepest gradient ascent. Taking the gradients of D with respect to the k_1-th and k_0-th prototypes we have

$$\nabla_{\mathbf{c}_1^{(k_1)}} D = \frac{2t}{k_1^{2/n}}(\mathbf{x}-\mathbf{c}_1^{(k_1)}) \tag{14a}$$

$$\nabla_{\mathbf{c}_0^{(k_0)}} D = -\frac{2t}{k_0^{2/n}}(\mathbf{x}-\mathbf{c}_0^{(k_0)}) \tag{14b}$$

In practice, we apply gradient ascent to all the $k=k_0+k_1$ nearest centroids. So, when a training sample $\mathbf{x}$ is taken, we use the following updating rules for its k nearest prototypes

$$\mathbf{c}_1^{(m_1)}(l+1) = \mathbf{c}_1^{(m_1)}(l) + \alpha\frac{2t}{m_1^{2/n}}(\mathbf{x}-\mathbf{c}_1^{(m_1)}(l)), \quad 1\le m_1\le k_1 \tag{15a}$$

$$\mathbf{c}_0^{(m_0)}(l+1) = \mathbf{c}_0^{(m_0)}(l) - \alpha\frac{2t}{m_0^{2/n}}(\mathbf{x}-\mathbf{c}_0^{(m_0)}(l)), \quad 1\le m_0\le k_0 \tag{15b}$$

where α stands for the learning rate and l represents the training iteration.

The kNN-LVC training phase starts with a set of labelled centroids. Then, the learning proceeds by sequentially taking training patterns and using equations (15a,b) to update the locations of the k nearest centroids to each pattern.

Note that the effective learning rate ($\alpha_e = 2\alpha t/m_i^{2/n}$) is different for every centroid: the furtherst prototype has the lowest α_e, what is used to compensate for the longest displacement vector ($\mathbf{x}-\mathbf{c}_i^{(k_i)}$).

Regarding selection of parameter k, it is made through an exploration process depending on factors like dimensionality and number of prototypes (as in the kNN methods). An additional aspect to consider when using a reduced number of centroids is that we should not use very high values for k, in order to keep fundamentals of the kNN procedures as local classifiers[1].

4 Experimental work

In this section we will check kNN-LVC performance by comparing it to other classifiers with a similar structure, including both trainable and non-trainable approaches. Let us begin with an outline of these classification schemes:

- kNN: labels a sample according to the majority class among its k nearest neighbors from the whole training set.
- Reduced kNN: is a kNN classifier applied to a reduced set of training samples, randomly selected among the training set. It involves lower computational and storage requirements in comparison to the previous classifier.
- LVQ1: is the simplest algorithm of the LVQ type. It also starts from a reduced set of prototypes, although this is an iterative learning method. In every learning stage, LVQ1 chooses a training sample and determines its closest prototype. This prototype is moved towards the sample if they belong to the same class; otherwise, it is moved away from the pattern. In both cases the displacement is proportional to the distance between the sample and the prototype. This proportionality factor α is the learning step for LVQ1. As for classification, LVQ1 assigns a sample to the same class as that of its nearest prototype.
- LVQ3 is an improved version of LVQ1, where the two nearest prototypes (say $\mathbf{c}_1$ and $\mathbf{c}_2$) to a sample $\mathbf{x}$ are simultaneously updated in the following manner:
  - if $\mathbf{x}$, $\mathbf{c}_1$ and $\mathbf{c}_2$ belong to the same class, these two prototypes are moved towards $\mathbf{x}$ a multiple ϵ of the learning rate α.
  - if just $\mathbf{c}_1$ or $\mathbf{c}_2$ has the same label as $\mathbf{x}$ and the sample falls into a zone, called window (w), around the midplane of $\mathbf{c}_1$ and $\mathbf{c}_2$, LVQ3 shifts the prototypes towards or away from $\mathbf{x}$, depending on whether prototype and sample have the same label or not, repectively.

  Note that this modification of LVQ1 is largely heuristic, in addition to require two additional parameters (ϵ and w) for the learning phase. On the other hand, classification with LVQ3 is identical to that of LVQ1.

It is interesting to remark that the proposed kNN-LVC can be seen as a generalization of the previous trainable approaches. So then, LVQ1 is equivalent to kNN-LVC when $k = 1$. Regarding LVQ3, we could consider that its training is a variant of the kNN-LVC learning when $k = 2$ and some restrictions are added. Nevertheless, the goal is different: LVQ-type methods seek to place prototypes in order to approximate the class distributions at the same time as achieving good accuracy rates, while kNN-LVC is only concerned about the latter.

[1] We consider techniques such as kNN to be local in the sense that the assignment of a sample to a class only depends on a near area around the sample, no matter what happens in further regions.

As for parameters used by the previous classifiers, we have explored different values for all of them, reporting the best results in our experiments. We indicate below the range of values we have considered for each parameter:

- number of prototypes: except for kNN, where all training samples are considered as prototypes, we have tested some values between 5% and 20% of the training set. However, when the best accuracy was achieved with 5% of the centroids, we further decreased this percentage until performance worsened. All methods start with the same prototypes, which are randomly chosen among the samples of each class.

- parameter k is only used in the case of kNN rules and kNN-LVC. Exploration range for k goes from 1 to the number of prototypes minus one.

- learning rate α: as suggested in [7], we have initialized it large enough to achieve fast adjustment and have gradually decreased it for fine-tuning. In our experiments we have used a linearly decreasing towards zero schedule, with 10 different initial values within $[0.005, 0.6]$.

- number of epochs: we have sequentially presented all training samples 60 times, having checked this is enough for all the methods to converge.

- regarding additional parameters of LVQ3, we have explored the ranges recommended in [7], testing three values for each one: $w = \{0.2, 0.25, 0.3\}$ and $\epsilon = \{0.1, 0.25, 0.5\}$.

We have applied these classification techniques to five pattern recognition problems extensively used as benchmarks. The first one, denominated "kwok", is a synthetic 2-dimensional dataset proposed in [8], with $500 - 10200$ (train$-$test) samples and having a 88.7% Bayessian classification limit. The rest are from the UCI Machine Learning Repository [2]: Waveform Data Generator ("waveform" in this paper) has 21 features and $4000 - 1000$ patterns; Image Segmentation ("image") is a 18 dimensional dataset with 1848 train and 462 test samples; Abalone ("abalone") has 8 dimensions and 2507-1670 samples; Pima Indians Diabetes ("diabetes") is a 8-dimensional dataset (768 samples) with no train$-$test partition, so in this case we have created 10 different partitions with a $60\% - 40\%$ train$-$test percentage, preserving in all partitions the prior probabilities. Excepting "waveform" and "diabetes", where approximately 1/3 of samples belong to C_1, the rest of problems have a quite similar number of patterns for both classes. In "waveform", "image" and "abalone" we have used the common binary version of the multiclass problem. We have preprocessed data so that all features have values within $[-1, 1]$. We should remark that all these problems have more than fifty training samples per dimension, what we think is enough so that we can apply kNN-type techniques.

Results in Table 1 show the average accuracy (in %) and standard deviation (in brackets) on the test set, obtained through 10 runs with different initial prototypes.

The corresponding parameters are displayed in Table 2. Except for the kNN rule, the first value indicates the percentage of the training set used as prototypes; parameter k in the first two columns and kNN-LVC denotes the number of nearest prototypes used to classify a sample; α is the initial value for the learning step in the trainable methods; finally, we also show parameters w and ϵ for LVQ3.

Table 1: Average classification accuracy (in %) and standard deviation (in brackets) of different classification methods. Results in boldface correspond to the best accuracy for each problem.

	kNN	Reduced kNN	LVQ1	LVQ3	kNN-LVC
kwok	**88.0**	86.9 (0.8)	87.8 (0.1)	87.8 (0.4)	**88.0** (0.1)
waveform	**92.1**	89.3 (1)	91.7 (0.4)	92.0 (0.2)	91.9 (0.4)
image	**97.4**	92.9 (1)	96.1 (0.5)	96.3 (0.5)	96.4 (0.4)
abalone	79.5	76.7 (1)	78.8 (0.4)	79.2 (0.5)	**79.7** (0.4)
diabetes	74.5	71.0 (1.9)	76.5 (3.1)	76.2 (3.1)	**77.0** (1.8)

Table 2: Parameters for the classification methods and benchmark problems.

	kNN	Reduced kNN	LVQ1	LVQ3	kNN-LVC
kwok	$k = 64$	20% $k = 13$	2.5% $\alpha = 0.6$	2.5%, $\alpha = 0.005$ $w = 0.25, \epsilon = 0.1$	10%, $k = 5$ $\alpha = 0.1$
waveform	$k = 57$	20% $k = 35$	0.4% $\alpha = 0.1$	0.4%, $\alpha = 0.005$ $w = 0.25, \epsilon = 0.25$	1.1%, $k = 3$ $\alpha = 0.01$
image	$k = 1$	20% $k = 1$	20% $\alpha = 0.3$	8%, $\alpha = 0.02$ $w = 0.3, \epsilon = 0.25$	20%, $k = 3$ $\alpha = 0.1$
abalone	$k = 39$	20% $k = 13$	4% $\alpha = 0.2$	4%, $\alpha = 0.02$ $w = 0.3, \epsilon = 0.1$	20%, $k = 9$ $\alpha = 0.02$
diabetes	$k = 29$	20% $k = 15$	3% $\alpha = 0.3$	2%, $\alpha = 0.3$ $w = 0.25, \epsilon = 0.25$	6.5%, $k = 3$ $\alpha = 0.1$

From these experiments, we observe that kNN and kNN-LVC get the best recognition rates. Both classifiers provide comparable results in all problems excepting "image" and "diabetes". In "image", the best kNN result is achieved considering only the nearest prototype, so it seems reasonable that reducing the number of prototypes and considering more than the nearest one will not lead to better accuracy. The case of "diabetes" is the opposite: kNN-LVC beats kNN, and this is because parameter k in kNN is quite high in relation to the dataset size. This fact allows that, even though the number of prototypes decrease considerably, it is possible to make a reliable estimate of the class membership considering more than the nearest centroid.

It is important to note that the classifier we propose requires less than 20% of the kNN prototypes, what implies kNN classification cost is very high in comparison to kNN-LVC. Furthermore, the number k of prototypes used to classify a sample is much lower in our approach, what is also advantageous if we apply techniques as [6] to reduce the searching time.

When the kNN method is applied over a reduced dataset (see 'Reduced kNN' in Table 1), it leads to a worsening in the recognition rates, what suggests that a learning phase for the prototypes' location is quite advisable.

Regarding the trainable schemes, we observe that kNN-LVC always outperforms LVQ1, probably because our method takes into account more than the nearest prototype. When we compare kNN-LVC with LVQ3, our approach usually achieves better

recognition rates. Additionally, the design of kNN-LVC is simpler than that of LVQ3, since our approach involves less parameters to adjust. As was to be expected, superiority of kNN-LVC is usually at the expense of a higher number of centroids.

From our experiments, we have noticed that the use of many centroids in LVQ-type methods can produce unwanted generalization effects. This is specially a problem when overlap is high, since there will be prototypes near the boundary that will produce misclassifications. We have checked kNN-LVC reduces this effect since the training samples in the overlap region push the centroids away from the boundary area, thus providing better generalization.

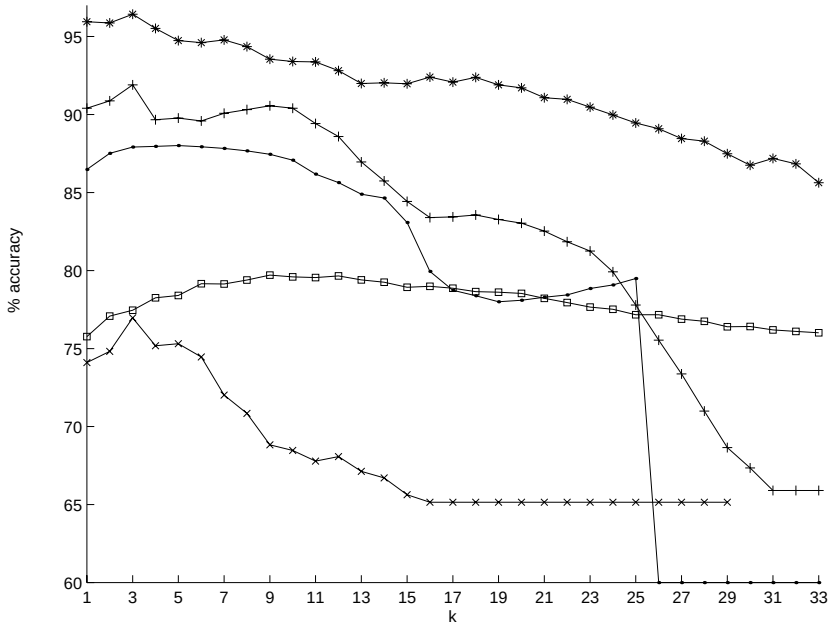

Figure 1: Evolution of the average accuracy rate as a function of parameter k for the kNN-LVC method. Represented problems are: "kwok" (dot marker), "waveform" (plus marker), "image" (star marker), "abalone" (square marker) and "diabetes" (cross marker).

In Figure 1 we illustrate the evolution of kNN-LVC accuracy when the proposed technique is trained with the parameters of Table 2 and different values for k. We observe that the general behavior (except for "image") is an initial increase in accuracy as we consider more prototypes, until their number is too high and the method loses its localized fundamentals. This explains the usual higher accuracy of kNN-LVC in comparison to LVQ-type techniques.

5 Conclusions and further work

In this paper we have presented the k Nearest Neighbor Learning Vector Classifier (kNN-LVC), a trainable scheme derived from a new variant of the kNN technique and the Bayes' decision rule for minimum error. We have shown that kNN-LVC is a simple method which preserves the fundamentals of kNN classifiers while solving the main problem of the kNN rule: its heavy storage and computational burden.

Quite similar to kNN-LVC as for architecture, interpretability and operation are the LVQ-type methods. In fact, we have shown that kNN-LVC can be considered as a generalization of LVQ when we remove the constraint of modelling the data distribution.

Nevertheless, the kNN-LVC's capacity to take into account k prototypes provides our method with more potential.

Experimental results through five benchmark problems have demonstrated the good performance of kNN-LVC in cases with enough number of training data in relation to dimensionality, both in comparison to kNN and LVQ techniques.

Regarding applications, we could further exploit the kNN-LVC gradient trainable structure and use it in non-static problems, leading to adaptive schemes of great interest in settings as speech processing or communication channels.

Ongoing research includes application of kNN-LVC in schemes with many initial prototypes (similar to dense clustering), as well as dealing with the outputs of ensembles of classifiers.

References

[1] C. M. Bishop, Neural Networks for Pattern Recognition. Oxford University Press, Oxford, UK, 1995.

[2] C. L. Blake and C. J. Merz, UCI Repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences, [http://www.ics.uci.edu/∼mlearn/MLRepository.html], 1998.

[3] T. Cover and P. Hart, Nearest Neighbor Pattern Classification. In *IEEE Transactions on Information Theory*, vol. **13**, no. 1, (1967) 21–27.

[4] R. Duda and P. Hart, Pattern Classification and Scene Analysis. John Wiley & Sons, 1973.

[5] K. Fukunaga, Introduction to Statistical Pattern Recognition. New York, NY: Academic Press, 2nd edn., 1990.

[6] K. Fukunaga and P. Narendra, A branch and bound algorithm for computing k-nearest neighbors, *IEEE Transactions on Computers* vol. **24** (1975) 750–753.

[7] T. Kohonen, The Self-Organizing MAP, *Proc. IEEE* vol. **78** (1990) 1464–1480.

[8] J. T. Kwok, Moderating the Output of Support Vector Machine Classifiers, *IEEE Transactions on Neural Networks* vol. **10**, no. 5 (1999) 1018–1031.