Pergamon

Neural Networks, Vol. 10, No. 6, pp. 1083–1088, 1997
© 1997 Elsevier Science Ltd. All rights reserved
Printed in Great Britain
0893–6080/97 $17.00+.00

PII: S0893-6080(97)00012-9

CONTRIBUTED ARTICLE

Learning Vector Quantization with Training Count (LVQTC)

ROBERTO ODORICO

University of Bologna, Istituto Nazionale di Fisica Nucleare

(*Received* 20 *November* 1995; *accepted* 2 *December* 1996)

Abstract—*Kohonen's learning vector quantization (LVQ) is modified by attributing training counters to each neuron, which record its training statistics. During training, this allows for dynamic self-allocation of the neurons to classes. In the classification stage training counters provide an estimate of the reliability of classification of the single neurons, which can be exploited to obtain a substantially higher purity of classification. The method turns out to be especially valuable in the presence of considerable overlaps among class distributions in the pattern space. The results of a typical application to high energy elementary particle physics are discussed in detail.* © *1997 Elsevier Science Ltd.*

Keywords—Learning vector quantization, Neural network architecture, Training, Classification, High energy physics, Elementary particle physics.

1. INTRODUCTION

Learning vector quantization (Kohonen, 1984, 1989, 1995) has proved over the years to represent a highly successful approach to pattern classification in a wide range of application fields. One should not be surprised, though, that as it is applied in a new field, presenting a different set of requirements, modifications are needed to meet new challenges.

High energy elementary particle physics has recently become a mature field for neural network applications. That is largely due to the availability of particle accelerators which have substantially increased the collision energy at which particle interactions can be studied. As a result, the problem of sorting out complex interaction events, in which several hundreds of particles are produced, has become rather common. Typically, one needs to select a small fraction of events associated with some signal of interest and to discard the remaining events, constituting the background. Neural network technology can represent a handy tool for this task. In particular, this kind of pattern classification problem can be handled by LVQ. However, for typical choices of the input variables, the signal and background class distributions in pattern space present strong overlaps which are hardly coped with by LVQ and its existing modifications. One is especially interested in selecting an event sample exhibiting a high degree of purity for the signal, paying willingly the price of substantial losses in the signal collection efficiency. In fact, the problem is not so much that of losing produced events but rather that of collecting a sample containing as much as possible only the good ones.

The modification to LVQ proposed here is meant to handle situations with strong overlaps among class distributions. For that sake, each LVQ neuron is provided with a set of training counters which keep track of the training statistics of the neuron. In particular, at the end of training, one can know how many times the neuron has been trained by patterns of its own class or of different classes. That allows us to estimate how reliable the classifications given by the neuron are. Thus, one can discard classifications provided by unreliable neurons like, e.g., those sitting in regions where the fraction of different class patterns is substantial. Neuron training counters can also be exploited during training, for self-allocation of neurons to classes, for neuron pruning and for neuron creation.

Section 2 summarizes the standard LVQ. In Section 3, the LVQTC algorithm is presented. Section 4 deals with a typical application in high energy elementary particle physics, and Section 5 contains the conclusions.

2. LEARNING VECTOR QUANTIZATION

Vector quantization amounts to a subdivision of the pattern space in populated regions (Voronoi tessellation) described by some suitably defined centroid vectors. Learning vector quantization (LVQ) is an algorithm (Kohonen, 1984) which adaptively builds up vector quantization from a training set of patterns. It can be

Requests for reprints should be sent to R. Odorico, Department of Physics, University of Bologna, Via Irnerio 46, I-40126 Bologna, Italy; Tel. and Fax: +39-51-242018; e-mail: odorico@bo.infn.it

used for the sake of data compression or, if more pattern classes are present, for pattern classification.

A set of class-labeled reference vectors m in the pattern space is introduced. They may be initially set equal to some training patterns s having their same classes. Then, the whole training set of class-labeled vectors $s(t)$, $t = 1, 2, 3\ldots$ is presented. For each $s(t)$ one finds the reference vector m_c closest to it, according to some distance definition. The position of m_c is updated: if $s(t)$ and m_c have the same class, then m_c is made closer to $s(t)$ by some amount proportional to their relative distance; otherwise m_c is moved away from $s(t)$ by a similar amount. The training set is recurrently read over, progressively reducing the amount of the corrections, until some stabilization is reached. Classification of a pattern vector s of unknown class is carried through by assigning the vector to the class of its closest reference vector.

Kohonen (1989, 1995) has subsequently proposed modifications, LVQ2 and LVQ3, meant to provide an improved performance near decision borders in pattern spaces. However, for strongly overlapping class distributions (and thus lack of definite decision borders) its motivations remain to be clarified. Modifications of LVQ2 which have been presented (e.g., Solaiman, Mouchot, & Maillard, 1994) do not settle this point.

3. LEARNING VECTOR QUANTIZATION WITH TRAINING COUNT

Learning vector quantization with training count (LVQTC) represents a modification of the original LVQ scheme, where additional attributes are appended to each neuron. The attributes record statistical information about the training undergone by the neuron. The aim of LVQTC is to classify pattern vectors s according to pattern classes $C_1, C_2, C_3\ldots C_N$. The additional neuron attributes are exploited both during training and classification. During training, they help to replace neurons with poor training performance and to create new neurons when they are needed. During classification, they provide an estimate of the reliability of the classification given by each neuron.

Neurons are defined by the following set of attributes:

a reference vector m in the pattern space;
a class label;
a counter for each class $P_1, P_2, P_3\ldots P_N$, storing the number of times vectors of that class have trained the neuron (training counters); and
a vector w in the pattern space, representing the centroid of wrong class (i.e., different from the neuron class) patterns which have trained the neuron.

Patterns used for training are certified to belong to the classes $C_1, C_2, C_3\ldots C_N$, without particular restrictions as to the numbers of patterns for each class. In the following, G_i and N_i will denote the global (or a priori) probability and the total number of training patterns of class C_i, respectively.

Neuron initialization makes use of the information on class distributions available from the class certified input patterns used to train the net. The initial total number of neurons is taken as a small fraction (about 5% or less) of the total number of input patterns. That is in order to limit the occurrence of neurons which will eventually be trained by statistically poor pattern samples. Neurons are allocated to classes proportionally to the volumetric and linear sizes of their distributions in pattern space. Class volume is estimated as the square root of the determinant of the corresponding correlation matrix (times a factor of 2 for each pattern variable), and the class linear size as twice the square root of its trace. (The justification for that is easy to see by considering the correlation matrix in diagonal form.) Neuron reference vectors m are initialized by training patterns of the same class, taken at random.

Training of neurons is arranged in a succession of epoches. For each epoch, the following steps are taken:

- neuron resetting: set all neuron training counters and ws to zero;
- neuron training: present the whole training set of class-labeled vectors $s(t)$, $t = 1, 2, 3\ldots$, choosing the pattern class at random each time. For each $s(t)$ find the neuron with the closest reference vector, m_c. Let the training vector $s(t)$ belong to class C_s, and suppose that m_c is labeled according to class C_c. Increment the training counter P_s of the neuron for class C_s by 1. Update m_c, leaving the other neuron reference vectors unchanged, according to:

$$m_c(t+1) = m_c(t) = (\alpha_r/P_{\text{tot}})[s(t) - m_c(t)] \text{ if } C_s = C_c \quad (1)$$

$$m_c(t+1) = m_c(t) - \alpha_w/P_{\text{tot}})[s(t) - m_c(t)] \text{ if } C_s \neq C_c$$

α_r and α_w are two distinct learning parameters. One should take: $\alpha_w, \alpha_r < 1$ (Kohonen, 1984). $P_{\text{tot}} = P_1 + P_2 + P_3\ldots + P_N$ is the current sum over all training counters of the m_c neuron. α_r and α_w are monotonically decreasing with the number of epoches: for each successive epoch they are reduced by a factor $F_r < 1$. If $C_s \neq C_c$, update the w of the m_c neuron;[1]
- neuron pruning and creation: after the presentation of the training set and before a new epoch is started, prune undertrained neurons and create new neurons to reduce neuron contamination from wrong class training patterns, according to the following rules applied in order:

[1] For w, one keeps a vector summer, where one adds all the $C_s \neq C_c$ $s(t)$ vectors training the neuron, and a scalar accumulator recording their number. At any moment, w can be calculated by dividing the vector summer by the value of the scalar accumulator. "Update the w" in practice means to add the $s(t)$ vector to the vector summer and to increase the scalar accumulator by one.

- neuron pruning: eliminate neurons with

$$P_{\mathrm{tot}} = P_1 + P_2 + \ldots + P_n < P_{\mathrm{prn}} \qquad (2)$$

where P_{prn} is a user-modifiable cutoff parameter (for a discussion of its value, see at the end of this section); and
- neuron creation: let us denote by P_w the maximum value of the wrong class training counters of the neuron; if $P_w > P_{\mathrm{prn}}$, create a neuron of a class whose training counter $P_i = P_w$ and with a vector m equal to the w of the original neuron.

Training stops when the number of right classifications on the training set no longer improves appreciably. For the calculations reported here, training has been stopped when for three successive epoches the number of right classifications on the training set is not larger than $(1 + F_{cvg})$ times the number of right classifications of the previous epoch, where $0 < F_{cvg} < 1$.

At the end of the last epoch, no neuron pruning and creation is made. Training counters are recalculated by reading the whole set of training patterns and keeping neurons frozen. At the same time, one calculates the neuron contamination by wrong class trainings, f_{cont}, and a neuron radius, R_{neu}, as follows.

f_{cont} is defined by:

$$f_{\mathrm{cont}} = A_x / A_{\mathrm{tot}} \qquad (3)$$

where

$$A_{\mathrm{tot}} = A_1 + A_2 + A_3 + \ldots + A_N \qquad (4)$$

with

$$A_i = G_i P_i / N_i \qquad (5)$$

G_i and N_i being the global probability and the total number of training patterns of class C_i, respectively, P_i being the training counter for class C_i of the neuron, and

$$A_x = \text{sum over all } A_i \text{ except } A_k \qquad (6)$$

k indicating the class C_k which labels the neuron.

R_{neu} is calculated as the square root of the mean square (r.m.s.) of the distance from the neuron of all the training patterns which have it as the closest neuron and belong to its same class.

The neuron parameters f_{cont} and R_{neu} are used, together with P_{tot}, in the classification process.

Classification of a pattern vector s of unknown class is carried through by assigning the vector to the class of its closest neuron. An estimate of the classification uncertainty is provided by the neuron contamination f_{cont}, introduced above. The classification may also be considered uncertain, if the neuron has too low a P_{tot} (i.e., its training is statistically poor) or is too distant from s. Then, the classification can be considered unreliable if for the closest neuron one has:

$$f_{\mathrm{cont}} > f_{\mathrm{cmax}} \qquad (7)$$

$$P_{\mathrm{tot}} < P_{\mathrm{min}}$$

$$D > D_{\mathrm{max}}$$

where f_{cmax}, P_{min} and D_{max} are user-defined cutoff parameters; D is the distance of s from the neuron measured in units of its R_{neu}.

In the following, there will be made reference to classification efficiency and purity. The classification efficiency ϵ is defined as the fraction of events in the original signal sample which are correctly classified as signal events. The classification purity π is defined as the fraction of actual signal events which are present in the sample of events classified as signal.

Distance metric can be chosen from a variety of options. Straightforward Euclidean metric is fast to calculate, but it may be inadequate if pattern variables range over widely different scales. A wiser choice is provided by a Euclidean metric weighted by the inverses of the pooled-over-classes variances of the single variables. In other words, one gauges the input variables in units of their variances over the training set. That does not cost a great loss in computing speed.

Two important consequences of keeping track of the training count for each neuron are:

(1) the number of neurons assigned to each class is no longer required to be proportional to the global probability for the class, as in standard LVQ. In the latter the density of neurons of each class in pattern space represents the only ingredient to (statistically) control classification in overlap regions. In LVQTC, on the other hand, this role is largely taken over by the training counters. That can be exploited by assigning relatively few neurons to classes which are concentrated in small regions of the pattern space, and more neurons to classes which are spread out over large regions. In this way, with a given total number of neurons, one can better represent the shapes of the class distributions; and

(2) similarly, the number of training patterns for each class is no longer required to be proportional to the global probability for the class, as in standard LVQ. As for neurons, one can more usefully allocate the total number of training patterns to classes according to the effective sizes of their distributions in pattern space, so as to have a more uniform neuron training. That may also be expedient when dealing with limited samples of training patterns for some, or all, classes: all training vectors can be exploited without having to leave part of them unused so as to satisfy the constraint of proportionality to class probabilities.

In LVQTC a fuller use is made of the information available on the distribution of training patterns, which is stored into the neuron training counters. Exploiting this information, neurons are left more sparse in regions of small class overlap and concentrated in regions of substantial class overlap. Training counters, together with neuron "radius" information, are also of help in estimating the reliability of pattern classifications, as discussed above.

On a general ground, LVQ can be viewed as a three-layered neural network: (1) the first, input, layer has as many neurons as the pattern variables; (2) the second, intermediate layer, contains the standard LVQ neurons; and (3) the third, output, layer has as many neurons as the number of classes. The intermediate neurons are governed by a winner-takes-all dynamics and compete for being corrected by the excitations of input neurons. Each one of them is connected to just one output neuron (according to its class label), whose excitation can be either 0 or 1. In LVQTC, the excitation of an output neuron can be identified with $1 - f_{\text{cont}}$, where f_{cont} is the contamination of the winning intermediate neuron firing into it, and thus it varies with continuity between 0 and 1. Also, the firing can be inhibited if the intermediate neuron has not received enough training or if the input pattern is too distant from it. An advantage with respect to other neural net architectures is that at the end of training the reference vectors of intermediate neurons can be directly interpreted as "typical" class patterns.

Overlaps between class distributions can be estimated from LVQTC. Let us consider the distributions $p_1(s)$ and $p_2(s)$ of classes C_1 and C_2, respectively, normalized to the corresponding global class probabilities G_1 and G_2. Let us define the overlap between the two distributions as

$$O(1,2) = 1 - \int |p_1(s) - p_2(s)| ds / | \int |p_1(s) + p_2(s)| ds \quad (8)$$

so that the overlap of a class distribution with itself is 1, and the overlap of two non-overlapping class distributions is 0). Then 0(1,2) can be estimated from the LVQTC neuron training counters as

$$0(1,2) = 1 - \sum\nolimits_n |P_1(n)G_1/N_1 - P_2(n)G_2/N_2| / [G_1 + G_2] \quad (9)$$

where the sum is extended over all neurons, $P_i(n)$ is the training counter of neuron n for class C_i, G_i is the global probability and N_i is the number of training patterns for class C_i.

The various LVQTC training and classification parameters should be varied to achieve optimal classification of an independent test set of class certified vectors. α_r and α_w can be increased to improve speed or decreased to refine convergence (initial values of about 0.1 are typically used). Decreasing P_{pm} will increase the number of neurons and thus resolution in pattern space, while increasing it will make neuron positions more reliable statistically and consequently will improve generalization. The total number of neurons can be increased to better represent details of the shapes of the class distributions. However, the number of neurons should be kept much smaller than the number of training patterns, to avoid that neurons simply act as a look-up table for the training patterns, which would hamper generalization. Also, a small number of neurons speeds up classification. The classification cutoff parameters f_{cmax}, P_{min} and D_{max} should be chosen according to how much purity one wishes to achieve in the classification process at the price of depriving classification efficiency. Parameter tuning can be realized by embedding LVQTC in a minimization program and choosing a suitable cost function (e.g., the number of misclassification on the test set of patterns). Each call from the minimization program is to be answered by a full training and classification LVQTC run. LVQTC training is typically fast enough to allow for that.

4. AN APPLICATION TO HIGH ENERGY PHYSICS

High energy elementary particle physics offers a host of examples of classification problems exhibiting a substantial overlap among classes. A typical problem consists in the extraction of a signal from a huge bulk of events strongly contaminated by background. The interaction between two elementary particles colliding at high energy usually leads to the production of a large number of elementary particles, which can reach up to a few hundreds. In a typical experiment, only a tiny fraction of the produced events (often hundreds of millions) are relevant: they have to be sorted out online (i.e., at the time one has to decide about their recording) by exploiting characteristic features of the multiparticle final state. This kind of "needle in the haystack" problem is solved by a multistage process in which a series a filters are employed: one first eliminates the most obvious background, and then one goes on trying to eliminate trickier and trickier backgrounds. One is rarely able to achieve a 100% pure signal and even more a 100% efficiency in its collection. One must rather make a tradeoff between classification purity and efficiency and try to increase as much as possible the signal/background ratio at the price of a limited loss for the signal collection efficiency. From what is measurable online on the final state it is not possible to draw a clear line separating signal from background: some background events are very close to signal events and sometimes undistinguishable from them as far as the allowed measurements are concerned. Neural networks can be of considerable help in this undertaking, since they can systematically exploit correlations among particle tracks and other measured quantities. However, they have to cope with the substantial overlap between signal and background events in the space of patterns of the measured variables presented to them for discrimination.

The example picked up here comes from an experiment at CERN aimed to the study of unstable ("beauty") particles produced in hadronic interactions (Baldanza et al., 1995). The particles decay before reaching the detectors. Thus, their production is proved by the presence, in addition to the primary interaction vertex to which most of the produced particle trajectories converge, of a secondary decay vertex to which the trajectories of particles generated in the decay converge. The problem is made more complicated by the fact that secondary vertices may also be due to the subsequent interaction of particles produced in the event with parts of the experimental apparatus. In order to make the signal discrimination amenable to a neural network approach, the large bandwidth raw data collected from the detectors have to be organized and condensed in a limited number of significant variables, constituting the input pattern to be submitted to the neural network. Details about that can be found in Baldanza et al. (1995). Here, it suffices to say that the input patterns are made of 14 variables, with integer values ranging from 0 to 16. The training set has been obtained by certification from a lengthy offline reconstruction and analysis of the experimental events. Three thousand training patterns have been used for the signal and the same amount for the background. The independent test set, for which results are quoted, has the same composition as the training set.

The LVQTC training converges in just 1 epoch ($F_{\mathrm{cvg}} = 0.013$), and leads to $N_s = 133$ neurons for the signal class and $N_b = 40$ neurons for the background class. The metric adopted is the Euclidean metric weighted by the inverses of the pooled-over-classes variances of the single variables. Training parameters have been initially set to the values: $\alpha_r = 0.2$, $\alpha_w = 0.1$, $F_r = 0.9$, $P_{\mathrm{pm}} = 10$. The volumes in pattern space of the signal and background distributions, V_s and V_b respectively, are in the ratio $V_s/V_b = 41.51$ as estimated from the correlation matrices, whereas their maximum linear sizes, L_s and L_b respectively, are in the ratio $L_s/L_b = 1.27$. That provides a rationale for why training requires a much higher number of neurons for the signal than for the background.

Accepting event classifications from all the neurons, one obtains on the test samples a classification purity $\pi = 0.69$, and a classification efficiency $\epsilon = 0.76$. By comparison, in standard LVQ, assigning 100 neurons to each class, one obtains $\pi = 0.72$, and $\epsilon = 0.66$ (convergence is achieved in two epoches).

In order to improve on purity, one can accept as proper classifications only those given by "reliable" neurons and discard the other ones as unreliable. The parameter we have used to estimate the neuron reliability is f_{cont}, gauging the fraction of times the neuron has been trained by patterns of a class different from its own [see eqn (3)]. In Figure 1 it is shown how purity and efficiency change when moving the upper cutoff on f_{cont}, referred to in the figure as "contamination cutoff". Purity increases substantially as the cutoff is brought down, reaching up to values of $\pi = 0.85$. Of course one has to pay a price in terms of efficiency, which, however, is kept at acceptable levels for the application at hand.

Also shown in Figure 1 are the corresponding results when using a neuron reliability measure based on distance. Specifically, denoting by D_c the distance of the classified pattern from the closest neuron and by D_x its distance from the closest neuron of class different from that of the classifying neuron, the ratio $D_c/(D_c + D_x)$ is considered and purity and efficiency are reported when varying the upper cutoff on this quantity. One might expect that purity would increase when reducing the cutoff on $D_c/(D_c + D_x)$. But in this application exactly the opposite occurs. That is related to the fact that the signal distribution is much wider than the background distribution and many of the signal neurons *sparsely* cover the vast regions with little background contamination. Patterns falling in these regions, although they could be reliably classified as signal, may easily have a relatively large $D_c/(D_c + D_x)$ and thus fall under the axe of the cutoff.

In Figure 2 purity is directly shown versus the corresponding efficiency when the cutoffs are varied, in

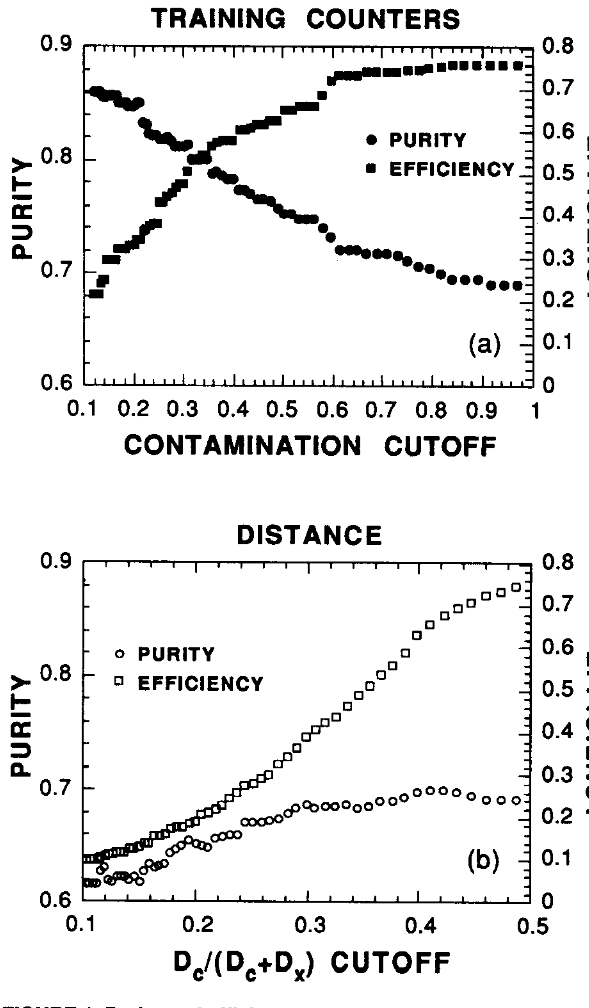

FIGURE 1. Purity and efficiency results versus the upper cutoffs on the neuron contamination f_{cont} (a) and on $D_c/(D_c + D_x)$ (b).

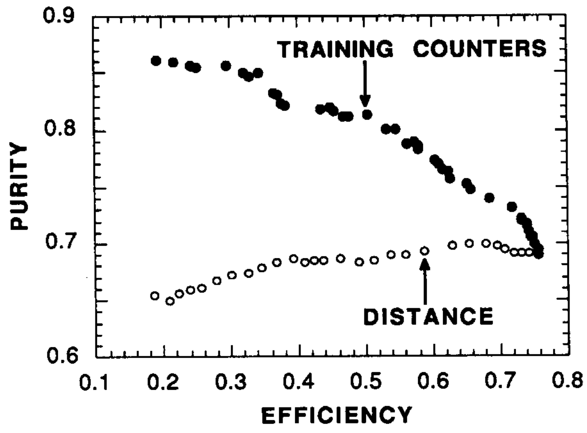

FIGURE 2. Same results as in Figure 1, with purity directly plotted versus efficiency.

order to better compare the results obtained when using the neuron training counters and the neuron pattern distance.

The present paper exclusively discusses the merits of the LVQTC modifications within LVQ. The interested reader, though, may find a comparison of the LVQTC results presented here with those from MLP and Fisher discrimination in Baldanza et al. (1995) (the enrichment factor ρ used there is related to the classification purity π introduced here by $1/\pi = 1 + 1/\rho$). A comparison in a different high energy physics problem among LVQTC, MLP and Fisher discrimination can be found in Mazzanti and Odorico (1993). In both applications the purity/efficiency performances of the three classifiers are qualitatively comparable.

5. CONCLUSIONS

The addition of neuron training counters to LVQ allows for a viable handling of classification problems with strongly overlapping class distributions in pattern space. In fact, it becomes possible to reject unreliable classifications and, thus, to select pattern samples with a considerably higher degree of class purity. In general, that cannot be achieved when trying to rely on criteria based on the neuron pattern distance. As a consequence, LVQTC can be used in a way similar to MLPs, with the MLP output replaced by the neuron contamination f_{cont} as the quantity gauging classification reliability. Training counters can also be exploited during training, so as to provide a dynamic allocation of neurons to classes by means of neuron pruning and creation.

REFERENCES

Baldanza, C., Bisi, F., Cotta-Ramusino, A., D'Antone, I., Malferrari, L., Mazzanti, P., Odorici, F., Odorico, R., Zuffa, M., Bruschini, C., Musico, P., Novelli, P., & Passaseo, M. (1995). Results from an on-line non-leptonic neural trigger in an experiment looking for beauty. *Nuclear Instruments and Methods*, **A361**, 506–518.

Kohonen, T. (1984). *Self-organization and associative memory* (2nd ed.). Berlin: Springer.

Kohonen, T. (1989). *Self-organization and associative memory* (3rd ed.). Berlin: Springer.

Kohonen, T. (1995). *Self-organizing maps*. Berlin: Springer.

Mazzanti, P., & Odorico, R. (1993). Bottom jet recognition by neural networks and statistical discriminants: a survey. *Zeitschrift für Physik*, **C59**, 273.

Solaiman, B., Mouchot, M. C., & Maillard, E. (1994). A hybrid algorithm (HLVQ) combining unsupervised and supervised learning approaches. In *Proceedings of IEEE International Conference on Neural Networks* (pp. 1772–1776). Orlando: IEEE.