

Machine learning in fMRI

Classifiers

Alexandre Savio, Maite Termenón, Manuel Graña

¹Computational Intelligence Group, University of the Basque Country

December, 2010



Outline

1 Motivation

- The classification problem
- Supervised Classification
- Unsupervised Classification

2 Available tools



Outline

1 Motivation

The classification problem

Supervised Classification

Unsupervised Classification

2 Available tools



The classification problem I

- In machine learning and pattern recognition, classification refers to an algorithmic procedure for assigning a given piece of input data into one of a given number of categories.



Names in Classification I

- An algorithm that implements classification, especially in a concrete implementation, is known as a classifier.
- The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.



Names in Classification II

- The piece of input data is formally termed an instance, and the categories are termed classes.
- The instance is formally described by a vector of features, which together constitute a description of all known characteristics of the instance.



Feature types

- Typically, features are either categorical (also known as nominal), consisting of one of a set of:
 - Unordered items
 - Ordinal items
 - Integer-valued items
 - Real-valued items
- Furthermore, many algorithms work only in terms of categorical data and require that real-valued or integer-valued data be discretized into groups (e.g. less than 5, between 5 and 10, or greater than 10).



Supervised learning I

- *Supervised learning* is based on determining a mapping between particular attributes, or features, of the data [2]
- A set of data points (*the training set*) is used to estimate (*learn*) the parameters of a model relating the features to the target labels.
- Once the parameters are learned, the model can be applied to predict the target label of a previously unseen data point.



Supervised learning II

- The supervised learning problem is referred to as *classification* when the target labels comprise a set of discrete classes, and as *regression* when the target labels assume continuous values.
- There are a number of different classification methods, each of which makes a different set of assumptions about the data and posits a particular type of model relating the features to the target labels, as well as a means of learning its parameters.



Supervised learning, classification and Clustering

- Classification normally refers to a supervised procedure, i.e. a procedure that learns to classify new instances based on learning from a training set of instances that have been properly labeled by hand with the correct classes.
- The corresponding unsupervised procedure is known as clustering, and involves grouping data into classes based on some measure of inherent similarity. .



Pattern recognition

- Classification and clustering are examples of the more general problem of pattern recognition, which is the assignment of some sort of output value to a given input value.
- Other examples are:
 - Regression, which assigns a real-valued output to each input;
 - Sequence labeling, which assigns a class to each member of a sequence of values
 - Parsing, which assigns a parse tree to an input sentence, describing the syntactic structure of the sentence
 - etc...



1 Motivation

The classification problem
Supervised Classification
Unsupervised Classification

2 Available tools



Supervised classification

- The process of using samples of known classes (training sets) and its labels to classify samples of unknown identity.



Common supervised classification algorithms

- Linear discriminant analysis (LDA)
- Artificial neural networks:
 - Multi-layer perceptron (MLP) trained with backpropagation
 - Probabilistic neural networks
 - Radial-basis function networks
- Learning vector quantization
- Support vector machines



Common supervised classification algorithms applied in fMRI

- Several commonly used linear classifiers in neuroimaging include:
 - linear support vector machine (SVM) [13, 17]
 - linear discriminant analysis (LDA) [17]
 - logistic regression (LR)
- There is no clearly “correct” choice of classifier for a given problem.
- LR and SVM are reported to have comparable performance [18], though SVMs can more efficiently handle high-dimensional feature spaces [24].



The two-class supervised classification problem I

- Given:
 - a set of training/testing input feature vectors
 $X = \{\mathbf{x}_i \in \mathbb{R}^n, i = 1, \dots, l\}$
 - the corresponding labels $\{y_i \in \{-1, 1\}, i = 1, \dots, l\}$, or $\mathbf{y} \in \{-1, 1\}^l$.
- The algorithms described next build some classifier systems based on this data.



The nearest neighbor approach I

- The simplest algorithm is the 1-NN which involves no adaptation and uses all the training data samples.
- The classification rule is of the form:

$$c(\mathbf{x}) = y_{i^*} \text{ where } i^* = \arg \min_{i=1, \dots, l} \{\|\mathbf{x} - \mathbf{x}_i\|\},$$

that is, the assigned class is that of the closest training vector.



Support Vector Machines I

- The Support Vector Machines (SVMs) have attracted attention from the pattern recognition community owing to a number of theoretical and computational merits derived from [24].
- SVM separates a given set of binary labelled training data with a hyperplane that is maximally distant from the two classes (known as the maximal margin hyperplane).
- The objective is to build a discriminating function using training data that will correctly classify new examples (\mathbf{x}, y) .



Support Vector Machines II

- When no linear separation of the training data is possible, SVMs can work effectively in combination with kernel techniques using the kernel trick, so that the hyperplane defining the SVMs corresponds to a nonlinear decision boundary in the input space that is mapped to a linearised higher-dimensional space [24].
- In this way the decision function can be expressed in terms of the support vectors only:

$$f(\mathbf{x}) = \text{sign} \left(\sum \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + w_0 \right)$$

where $K(.,.)$ is a kernel function, α_i is a weight constant derived from the SVM process and the \mathbf{s}_i are the support vectors [24].



Support Vector Machines III

- Given training vectors $\mathbf{x}_i \in \mathbb{R}^n, i = 1, \dots, l$ of the subject features of the two classes, and a vector $\mathbf{y} \in \mathbb{R}^l$ such that $y_i \in \{-1, 1\}$ labels each subject with its class, in our case, for example, patients were labeled as -1 and control subject as 1. [4]
- To construct a classifier, the SVM algorithm solves the following optimization problem:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^l \xi_i$$

subject to $y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq (1 - \xi_i), \xi_i \geq 0, i = 1, 2, \dots, n$. The dual optimization problem is

$$\min_{\alpha} \frac{1}{2} \alpha^T \mathbf{Q} \alpha - \mathbf{e}^T \alpha$$

subject to $\mathbf{y}^T \alpha = 0, 0 \leq \alpha_i \leq C, i = 1, \dots, l$, where \mathbf{e} is the vector of all ones, $C > 0$ is the upper bound on the error, \mathbf{Q} is an $l \times l$ positive semi-definite matrix, $Q_{ij} \equiv y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, and $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ is the kernel function that describes the behavior of the support vectors.



Support Vector Machines IV

- Here, the training vectors \mathbf{x}_i are mapped into a higher (maybe infinite) dimensional space by the function $\phi(\mathbf{x}_i)$.
- C is a regularization parameter used to balance the model complexity and the training error.



Support Vector Machines V

- The kernel function chosen results in different kinds of SVM with different performance levels, and the choice of the appropriate kernel for a specific application is a difficult task. In this study two different kernels were tested: the linear and the radial basis function (RBF) kernel.
- The linear kernel function is defined as $K(\mathbf{x}_i, \mathbf{x}_j) = 1 + \mathbf{x}_i^T \mathbf{x}_j$, this kernel shows good performance for linearly separable data.
- The RBF kernel is defined as $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$. This kernel is basically suited best to deal with data that have a class-conditional probability distribution function approaching the Gaussian distribution.



Support Vector Machines VI

- One of the advantages of the RBF kernel is that given the kernel, the number of support vectors and the support vectors are all automatically obtained as part of the training procedure, i.e., they do not need to be specified by the training mechanism.



Multi-layer perceptron trained with Backpropagation I

- Backward propagation of errors, or backpropagation (BP), [20, 10, ?] is a non-linear generalization of the squared error gradient descent learning rule for updating the weights of artificial neurons in a single-layer perceptron, generalized to feed-forward networks, also called Multi-Layer Perceptron (MLP).
- Backpropagation requires that the activation function used by the artificial neurons (or "nodes") is differentiable with its derivative being a simple function of itself.
- The backpropagation of the error allows to compute the gradient of the error function relative to the hidden units.
- It is analytically derived using the chain rule of calculus.



Multi-layer perceptron trained with Backpropagation II

- During on-line learning, the weights of the network are updated at each input data item presentation.
- We have used the resilient backpropagation, which uses only the derivative sign to perform the weight updating.
- We restrict our presentation of BP to train the weights of the MLP for the current two class problem.
- Let the instantaneous error E_p be defined as:

$$E_p(\mathbf{w}) = \frac{1}{2} (y_p - z_K(\mathbf{x}_p))^2, \quad (1)$$

where y_p is the p -th desired output y_p , and $z_K(\mathbf{x}_p)$ is the network output when the p -th training exemplar x_p is inputted to the MLP composed of K layers, whose weights are aggregated in the vector \mathbf{w} . The output of the j -th node in layer k is given by:



Multi-layer perceptron trained with Backpropagation III

$$z_{k,j}(\mathbf{x}_p) = f \left(\sum_{i=0}^{N_{k-1}} w_{k,j,i} z_{k-1,i}(\mathbf{x}_p) \right), \quad (2)$$

where $z_{k,j}$ is the output of node j in layer k , N_k is the number of nodes in layer k , $w_{k,j,i}$ is the weight which connects the i -th node in layer $k-1$ to the j -th node in layer k , and $f(\cdot)$ is the sigmoid nonlinear function, which has a simple derivative:

$$f'(\alpha) = \frac{df(\alpha)}{d\alpha} = f(\alpha)(1 - f(\alpha)). \quad (3)$$

The convention is that $z_{0,j}(\mathbf{x}_p) = \mathbf{x}_{p,j}$. Let the total error E_T be defined as follows:



Multi-layer perceptron trained with Backpropagation IV

$$E_T(\mathbf{w}) = \sum_{p=1}^l E_p(\mathbf{w}), \quad (4)$$

where l is the cardinality of X . Note that E_T is a function of both the training set and the weights in the network. The backpropagation learning rule is defined as follows:

$$\Delta w(t) = -\eta \frac{\partial E_p(\mathbf{w})}{\partial w} + \alpha \Delta w(t-1), \quad (5)$$

where $0 < \eta < 1$, which is the learning rate, the momentum factor α is also a small positive number, and w represents any single weight in the network. In the above equation, $\Delta w(t)$ is the change in the weight computed at time t . The momentum term is sometimes used ($\alpha \neq 0$) to improve the smooth convergence of the



Multi-layer perceptron trained with Backpropagation V

algorithm. The algorithm defined by equation (5) is often termed as *instantaneous backpropagation* because it computes the gradient based on a single training vector. Another variation is *batch backpropagation*, which computes the weight update using the gradient based on the total error E_T .

To implement this algorithm we must give an expression for the partial derivative of E_p with respect to each weight in the network. For an arbitrary weight in layer k this can be written using the Chain Rule:

$$\frac{\partial E_p(\mathbf{w})}{\partial w_{k,j,i}} = \frac{\partial E_p(\mathbf{w})}{\partial z_{k,j}(\mathbf{x}_p)} \frac{\partial z_{k,j}(\mathbf{x}_p)}{\partial w_{k,j,i}}. \quad (6)$$

Because the derivative of the activation function follows equation 3, we get:



Multi-layer perceptron trained with Backpropagation VI

$$\frac{\partial z_{k,j}(\mathbf{x}_p)}{\partial w_{k,j,i}} = z_{k,j}(\mathbf{x}_p) (1 - z_{k,j}(\mathbf{x}_p)) z_{k-1,j}(\mathbf{x}_p), \quad (7)$$

and

$$\frac{\partial E_p(\mathbf{w})}{\partial z_{k,j}(\mathbf{x}_p)} = \sum_{m=1}^{N_{k+1}} \frac{\partial E_p(\mathbf{w})}{\partial z_{k+1,m}(\mathbf{x}_p)} z_{k+1,m}(\mathbf{x}_p) (1 - z_{k+1,m}(\mathbf{x}_p)) w_{k+1,m,j},$$

which at the output layer corresponds to the output error :

$$\frac{\partial E_p(\mathbf{w})}{\partial z_k(\mathbf{x}_p)} = z_L(\mathbf{x}_p) - y_p. \quad (8)$$



Radial-basis function network I

Radial Basis Function networks (RBF) [5, 10] are a type of ANN that use radial basis functions as activation functions. RBFs consist of a two layer neural network, where each hidden unit implements a radial activated function. The output units compute a weighted sum of hidden unit outputs. Training consists of the unsupervised training of the hidden units followed by the supervised training of the output units' weights. RBFs have their origin in the solution of a multivariate interpolation problem [?]. Arbitrary function $g(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$ can be approximated by a map defined by a RBF network with a single hidden layer of K units:

$$\hat{g}_{\theta}(\mathbf{x}) = \sum_{j=1}^K w_j \phi(\sigma_j, \|\mathbf{x} - \mathbf{c}_j\|), \quad (9)$$

where θ is the vector of RBF parameters including $w_j, \sigma_j \in \mathbb{R}$, and $\mathbf{c}_j \in \mathbb{R}^n$; let us denote $\mathbf{w} = (w_1, w_2, \dots, w_p)^T$, then the vector of



Radial-basis function network II

RBF parameters can be expressed as

$\theta^T = (\mathbf{w}^T, \sigma_1, \mathbf{c}_1^T, \dots, \sigma_K, \mathbf{c}_K^T)$. Each RBF is defined by its center $\mathbf{c}_j \in \mathbb{R}^n$ and width $\sigma_j \in \mathbb{R}$, and the contribution of each RBF to the network output is weighted by w_j . The RBF function $\phi(\cdot)$ is a nonlinear function that monotonically decreases as \mathbf{x} moves away from its center \mathbf{c}_j . The most common RBF used is the isotropic Gaussian:

$$\hat{g}_\theta(\mathbf{x}) = \sum_{j=1}^p w_j \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2\sigma_j^2}\right).$$

The network can be thought as the composition of two functions $\hat{g}_\theta(\mathbf{x}) = W \circ \Phi(\mathbf{x})$, the first one implemented by the RBF units $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}^K$ performs a data space transformation which can be a dimensionality reduction or not, depending on whether $K > n$. The second function corresponds to a single layer linear Perceptron $W: \mathbb{R}^K \rightarrow \mathbb{R}$ giving the map of the RBF transformed data into the



Radial-basis function network III

class labels. Training is accordingly decomposed into two phases. First a clustering algorithm is used to estimate the Gaussian RBF parameters (centres and variances). Afterwards, linear supervised training is used to estimate the weights from the hidden RBF to the output. In order to obtain a binary class label output, a hard limiter function is applied to the continuous output of the RBF network.



Probabilistic neural networks I

A Probabilistic Neural Network (PNN) [23] uses a kernel-based approximation to form an estimate of the probability density function of categories in a classification problem. In fact, it is a generalization of the Parzen windows distribution estimation, and a filtered version of the 1-NN classifier. The distance of the input feature vector \mathbf{x} to the stored patterns is filtered by a RBF function. Let us denote the data sample partition as $X = X_1 \cup X_{-1}$, where $X_1 = \{\mathbf{x}_1^1, \dots, \mathbf{x}_{n_1}^1\}$ and $X_{-1} = \{\mathbf{x}_1^{-1}, \dots, \mathbf{x}_{n_{-1}}^{-1}\}$. That is, superscripts denote the class of the feature vector and $n_1 + n_{-1} = n$. Each pattern \mathbf{x}_j^i of training data sample is interpreted as the weight of the j -th neuron of the i -th class. Therefore the response of the neuron is computed as the probability of the input feature vector according to a Normal distribution centered at the stored pattern:

$$\Phi_{i,j}(\mathbf{x}) = \frac{1}{(2\pi)^{n/2} \sigma^n} \exp \left[-\frac{\|\mathbf{x} - \mathbf{x}_j^i\|^2}{2\sigma^2} \right], \quad (10)$$



Probabilistic neural networks II

Therefore the output of the neuron is inside $[0, 1]$. The tuning of a PNN network depends on selecting the optimal sigma value of the spread σ of the RBF functions, which can be different for each class. In this paper an exhaustive search for the optimal spread value in the range $(0, 1)$ for each training set has been done. The output of the PNN is an estimation of the likelihood of the input pattern \mathbf{x} being from class $i \in \{-1, 1\}$ by averaging the output of all neurons that belong to the same class:

$$p_i(\mathbf{x}) = \frac{1}{n_i} \sum_{j=1}^{n_i} \Phi_{i,j}(\mathbf{x}). \quad (11)$$

The decision rule based on the output of all the output layer neurons is simply:

$$\hat{y}(\mathbf{x}) = \arg \max_i \{p_i(\mathbf{x})\}, \quad i \in \{-1, 1\}. \quad (12)$$



Probabilistic neural networks III

where $\hat{y}(\mathbf{x})$ denotes the estimated class of the pattern \mathbf{x} . If the a priori probabilities for each class are the same, and the losses associated with making an incorrect decision for each class are the same, the decision layer unit classifies the pattern \mathbf{x} in accordance with the optimal Bayes' rule.



Learning vector quantization I

Learning Vector Quantization (LVQ) [11, 22] as introduced by Kohonen [12] represents every class $c \in \{-1, 1\}$ by a set $W(c) = \{\mathbf{w}_i \in \mathbb{R}^n; i = 1, \dots, N_c\}$ of weight vectors (prototypes) which tessellate the input feature space. Let us denote W the union of all prototypes, regardless of class. If we denote c_i the class the weight vector $\mathbf{w}_i \in W$ is associated with, the decision rule that classifies a feature vector \mathbf{x} is as follows:

$$c(\mathbf{x}) = c_{i^*}$$

where

$$i^* = \arg \min_i \{\|\mathbf{x} - \mathbf{w}_i\|\}.$$

The training algorithm of LVQ aims at minimizing the classification error on the given training set, i.e., $E = \sum_j (y_j - c(\mathbf{x}_j))^2$, modifying



Learning vector quantization II

the weight vectors on the presentation of input feature vectors. The heuristic weight updating rule is as follows:

$$\Delta \mathbf{w}_{i^*} = \begin{cases} \varepsilon \cdot (\mathbf{x}_j - \mathbf{w}_{i^*}) & \text{if } c_{i^*} = y_j \\ -\varepsilon \cdot (\mathbf{x}_j - \mathbf{w}_{i^*}) & \text{otherwise} \end{cases}, \quad (13)$$

that is, the input's closest weight is adapted either toward the input if their classes match, or away from it if not. This rule is highly unstable, therefore, the practical approach consists in performing an initial clustering of each class data samples to obtain an initial weight configuration using equation 13 to perform the fine tuning of the classification boundaries. This equation corresponds to a LVQ1 approach. The LVQ2 approach involves determining the two input vector's closest weights. They are moved toward or away the input according to the matching of their classes.



Combination of classifiers: AdaBoost I

- Adaptive Boosting (AdaBoost)[21, 8] is a meta-algorithm for machine learning that can be used in conjunction with many other learning algorithms to improve their performance.
- AdaBoost is adaptive in the sense that subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. Otherwise, it is less susceptible to the over-fitting problem than most learning algorithms.
- AdaBoost calls a weak classifier repeatedly in a series of rounds $t = 1, \dots, T$. For each call a distribution of weights W_t is updated and indicates the importance of examples in the data set for the classification.



Combination of classifiers: AdaBoost II

- On each round, the weights of each incorrectly classified example are increased (or alternatively, the weights of each correctly classified example are decreased), so that the new classifier focuses more on those examples.
- Following these ideas, we have also tested a combination of SVM classifiers following the the Diverse-AdaBoost-SVM [?].
 - In this approach we built a sequence of SVM classifiers of increasing variance parameter.
 - The results of the classifiers are weighted according to their statistical error to obtain the response to the test inputs in the 10-fold validation process.



1 Motivation

The classification problem
Supervised Classification
Unsupervised Classification

2 Available tools



Unsupervised classification

- The identification of natural groups, or structures/patterns, within data.
- Clustering is one form of unsupervised learning, which identifies groups within the data based on the similarity of their features.



Common unsupervised classification methods

- Most common algorithms:
 - k-nearest neighbors (k-NN)
 - k-means
 - Self organizing maps (SOM)
 - Hierarchical clustering



Common unsupervised classification applied in fMRI studies I

- **K-means clustering**
 - assigns each data point to one of k groups, and has been applied to discover relationships among the time series of voxels in fMRI data. [16]



Common unsupervised classification applied in fMRI studies II

- **Hierarchical clustering** methods [7, 14]
 - build a succession of clusters:
 - data-points are first grouped into clusters,
 - and the clusters themselves are merged into groups at a second level according to their similarity,
 - and so forth, building a tree depicting the hierarchical dependence structure across data points.



Common unsupervised classification applied in fMRI studies III

- Thus, while:
 - k-means clustering is informative of major subdivisions in the data.
 - hierarchical clustering provides a more complete characterization of relationships between data points and may be used to identify more subtle patterns.



Common unsupervised classification applied in fMRI studies IV

- However, some of the relationships discovered by hierarchical clustering:
 - may be driven by attributes specific to the dataset to which it is applied,
 - and may not generalize across other datasets.



Independent Component Analysis (ICA) I

- Independent component analysis (ICA) is another popular form of unsupervised learning. [6]
- ICA can be applied to decompose a set of fMRI time courses into a set of spatially distinct “networks”.
- ICA has provided insight into the functional organization of the brain [1] and suggested key functional differences across clinical populations [19].



Independent Component Analysis (ICA) II

- In addition to exploring the spatiotemporal structure of brain activity within individuals and groups,
 - the spatial networks and time courses derived using ICA can also be used to derive features within the framework of supervised learning [15].



Independent Component Analysis (ICA) III

- Calhoun et al. (2008) [3] applied ICA to the data of bipolar and schizophrenic patients, extracting two networks from each subject (“default-mode” and “temporal lobe”).
 - The networks were then used as input to a classification algorithm, which demonstrated high accuracy in classifying between the patient groups.



Lattice Independent Component Analysis (LICA) I

- The lattice sources discovered by the ILSIA are equivalent to the GLM design matrix columns, and the unmixing process is identical to the conventional least squares estimator.
- Therefore, LICA is a kind of unsupervised GLM whose regressor functions are mined from the input dataset.
- If we try to establish correspondences to the ICA, the lattice sources correspond to the unknown statistically independent sources and the mixing matrix is the one given by the abundance coefficients computed by least squares estimation.

[9]



Available tools

- Quick-R
- PyMVPA
- LibSVM
- SVM-light
- Weka

Software para clasificación (GIC Wiki)



References I



Christian F Beckmann, Marilena DeLuca, Joseph T Devlin, and Stephen M Smith.

Investigations into resting-state connectivity using independent component analysis.

Philosophical Transactions of the Royal Society B: Biological Sciences, 360(1457):1001 –1013, May 2005.






Signe Bray.

Applications of multivariate pattern classification analyses in developmental neuroimaging of healthy and clinical populations.

Frontiers in Human Neuroscience, 3, 2009.






References II

-  Vince D. Calhoun, Paul K. Maciejewski, Godfrey D. Pearlson, and Kent A. Kiehl.
Temporal lobe and “default” hemodynamic brain modes discriminate between schizophrenia and bipolar disorder.
Human Brain Mapping, 29(11):1265–1275, 2008.
-  Chih-Chung Chang and Chih-Jen Lin.
LIBSVM: a library for support vector machines.
2001.
Software available at
<http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
-  S. Chen, C.F.N. Cowan, and P.M. Grant.
Orthogonal least squares learning algorithm for radial basis function networks.
Neural Networks, IEEE Transactions on, 2(2):302–309, 1991.



References III

-  Pierre Comon.
Independent component analysis, a new concept?
Signal Processing, 36(3):287–314, April 1994.
-  Dietmar Cordes, Vic Haughton, John D. Carew, Konstantinos Arfanakis, and Ken Maravilla.
Hierarchical clustering to measure connectivity in fMRI resting-state data.
Magnetic Resonance Imaging, 20(4):305–317, May 2002.
-  Yoav Freund and Robert Schapire.
A decision-theoretic generalization of on-line learning and an application to boosting.
In *European Conference on Computational Learning Theory*, pages 37, 23, 1995.



References IV



Manuel Graña, Maite García-Sebastián, and Carmen Hernández.

Lattice independent component analysis for fMRI analysis.
In *Artificial Neural Networks - ICANN 2009*, volume 5769 of *Lecture Notes in Computer Science*, pages 725–734–734.
Springer Berlin / Heidelberg, 2009.



Simon Haykin.

Neural Networks: A Comprehensive Foundation.
Prentice Hall, 2 edition, July 1998.






T. Kohonen.

Self-organization and associative memory: 3rd edition.
Springer-Verlag New York, Inc., 1989.





References V

-  T. Kohonen.
Learning vector quantization.
In The handbook of brain theory and neural networks, pages
537–540. MIT Press, 1998.
-  Stephen LaConte, Stephen Strother, Vladimir Cherkassky, Jon
Anderson, and Xiaoping Hu.
Support vector machines for temporal classification of block
design fMRI data.
NeuroImage, 26(2):317–329, June 2005.
-  Wei Liao, Huafu Chen, Qin Yang, and Xu Lei.
Analysis of fMRI data using improved Self-Organizing mapping
and Spatio-Temporal metric hierarchical clustering.
Medical Imaging, IEEE Transactions on, 27(10):1472–1483,
2008.





References VI

-  Federico De Martino, Francesco Gentile, Fabrizio Esposito, Marco Balsi, Francesco Di Salle, Rainer Goebel, and Elia Formisano.
Classification of fMRI independent components using IC-fingerprints and support vector machine classifiers.
NeuroImage, 34(1):177–194, January 2007.
-  Aviv Mezer, Yossi Yovel, Ofer Pasternak, Tali Gorfine, and Yaniv Assaf.
Cluster analysis of resting-state fMRI time series.
NeuroImage, 45(4):1117–1125, May 2009.



References VII

-  Janaina Mourão-Miranda, Arun L.W. Bokde, Christine Born, Harald Hampel, and Martin Stetter.
Classifying brain states and determining the discriminating activation patterns: Support vector machine on functional MRI data.
NeuroImage, 28(4):980–995, December 2005.
-  Francisco Pereira, Tom Mitchell, and Matthew Botvinick.
Machine learning classifiers and fMRI: a tutorial overview.
NeuroImage, 45(1, Supplement 1):S199–S209, March 2009.






References VIII

-  Serge A.R.B. Rombouts, Jessica S. Damoiseaux, Rutger Goekoop, Frederik Barkhof, Philip Scheltens, Stephen M. Smith, and Christian F. Beckmann.
Model-free group analysis shows altered BOLD FMRI networks in dementia.
Human Brain Mapping, 30(1):256–266, 2009.
-  D. E. Rumelhart, G. E. Hinton, and R. J. Williams.
Learning internal representations by error propagation, pages 318–362.
MIT Press, 1986.
-  Robert E. Schapire and Yoram Singer.
Improved boosting algorithms using confidence-rated predictions.
Machine Learning, 37(3):297–336, December 1999.



References IX

-  P. Somervuo and T. Kohonen.
Self-Organizing maps and learning vector quantization for
feature sequences.
Neural Process. Lett., 10(2):151–159, 1999.
-  Donald F. Specht.
Probabilistic neural networks.
Neural Netw., 3(1):109–118, 1990.
-  V.N. Vapnik.
Statistical Learning Theory.
Wiley-Interscience, September 1998.

