Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Unlabeled pattern management through Semi-Supervised classification techniques

Relatore: Prof. Loris Nanni

Laureando: Giordano Segato

Corso di laurea magistrale in ingegneria informatica

Data di laurea: 14/10/2014

Anno accademico 2013/2014

To my family

Summary

The purpose of this thesis work is to analyze different state-of-art semi-supervised learning algorithms that has been proposed in last years. In semi-supervised learning, unlabeled data are used together with labeled data to provide better performance in the classification process.

In the experimental phase, the implemented algorithms have been applied to various datasets usually tested in the academic research. Used datasets are real-world datasets available at UCI (University of California, Irvine) website.

A second phase of the experiments consists in the fusion and combination of the analyzed algorithms in order to obtain a classification algorithm with good performance in each tested dataset. This goal represents the implementation of an algorithm with robustness properties, which can be used in a dataset independently from its nature, in a general purpose classifier.

Index

Chapter 1

Introduction

In the current Information Age there has been an extraordinary growth of data that have been generated and stored in millions of data structures. Consequently, the availability of very large volumes of data has created a problem of how to extract useful information.

Traditionally, data analysis techniques that have been used for such tasks include regression analysis, cluster analysis, numerical taxonomy, multidimensional analysis, other multivariate statistical methods, stochastic models, time series analysis, non-linear estimation techniques, and others. These techniques have been widely used for solving many practical problems.

Statistical data analysis is primarily oriented toward the extraction of quantitative data characteristics, and as such has inherent limitations. For example, a statistical analysis can determine average and correlations between variables in data, but it cannot characterize the dependencies at an abstract and conceptual level, providing a causal explanation of the reasons why these dependencies exist.

Moreover, the statistical techniques can determine the central tendency and variance of given factors, while regression analysis can fit a curve to a set of samples. However, these techniques can't produce a qualitative description of the dataset structure and determine the dependences not explicitly provided in the data.

Methods based on numerical analysis can create a classification of samples and specify a numerical similarity among the samples assembled into the same or different categories. This approach however can't build qualitative description of the reasons behind class assignment.

In summary, traditional data analysis techniques facilitate useful data interpretations and can help to generate important insights into the processes behind the data. In efforts to satisfy the growing need for new data analysis tools that will overcome the limitations of traditional statistical analysis, researchers have turned to ideas and methods developed in machine learning.

The last decade has experienced a revolution in terms of information availability and exchange. The World Wide Web and the amount of produced data is growing at an exponential rate.

Moreover many businesses and organizations have begun to collect data regarding their own operations and market opportunities on a large scale. An important challenge consists in the "extraction" of useful information from the provided amount of data. Beyond the immediate purpose of tracking or archiving the activities of an organization, the collected data can sometimes represent an important resource for strategic planning and decisions. Research and development in this area are often referred to as *data mining* and *knowledge discovery in databases (KDD)* [14].

The goal of data mining algorithms is to extract useful information from large data archives. Reached information can be obtained:

- *Directly*, in the form of "knowledge" characterizing the relations between the variables of interest.
- *Indirectly*, as functions that allow to predict, classify, or represent information in the distribution of the data.

In the field of data mining and knowledge discovery, new techniques and algorithms have been developed to deal with the computational complexity deriving from the large amount of data. However the large availability of data can provide good performance in the classification, also considering unlabeled data in the training phase, using techniques called semi-supervise learning [2] [10].

Many different approaches and algorithms have been proposed. For example techniques oriented to obtain robustness property (especially in the case of noisy datasets), or feature selection techniques which provide good performance with high-dimensional datasets [8].

In this thesis work, four algorithms have been implemented and analyzed, Analyzed methods have been published in papers distributed in the last years [6] [7] [8] [9]. Then the algorithms have been tested on various datasets available online in the UCI website: *https://archive.ics.uci.edu/ml/machine-learning-databases*. A detailed description of datasets is illustrated in sections 4.2 and 4.3.

The implemented algorithms regard different techniques and are described in the relative reference articles:

- Classifier which uses a generated set of artificial samples [6]
- Advanced technique based on semi-supervised learning [7]
- A semi-supervised feature ranking method [8]
- Using clustering analysis to improve semi-supervised classification [9]

These different approaches are described in section 3.

In section 4, the experiments that have been done are explained and the dataset division (labeled, unlabeled, validation data) is illustrated.

Chapter 2

Machine Learning

Machine Learning is the discipline that study algorithms which allow to identify patterns in data.

Over the past years Machine Learning has become one of the most important fields of information technology. With the ever increasing amounts of available data, there are good reasons to believe that smart data analysis will become even more pervasive as a necessary component of the technological progress.

Usually, major machine learning problems can be divided in three groups of problems: Association analysis, clustering and classification.

Association Analysis: this task consists in searching for patterns that describe strong associations among the features. The discovered patterns are typically represented in the form of implication rules between feature subsets. Because of the exponential size of its search space, the goal of association analysis is to extract the most interesting patterns in an efficient manner. Useful applications of association analysis include finding groups of genes that have related functionality, identifying web pages that are accessed together, or understanding the relationships between different elements of Earth's climate system.

Classification: it consists in the construction of a model that allows to classify records assigning them a label. The base form of classification has two phases: a learning phase which create a model

on a dataset of already classified data, and a test phase which is used to test the obtained model and get a measure of performance. Test phase tests the classifier built in training phase on a test set; the classifier assigns to any record a label and then resulting labels are compared with real classes and an accuracy measure is calculated.

Clustering: it is the problem of dividing data in clusters, based on the attributes values. It can be used for group newspaper articles or website and can be similar to classification for some aspects.
In this thesis will be used classification techniques prevalently based on the Support Vector Machine (SVM) classifier, so in the next this argument will be examined in depth.

2.1 Classification

Classification is the task of assigning a label to an input object. It's a very common problem and it has many different real-world applications. Examples include detecting spam email messages based on the message header and content, categorizing cells as malignant or the classification of galaxies based upon their shape.
To solve the classification problem, different learning algorithms have been proposed and each algorithm has advantage and disadvantage. Most diffused algorithms are:

Decision Tree: it is very simple to implement and it has good time performance. However accuracy is not very high and there are more convenient learning algorithms. It consists in the constructions of a decision tree by progressively splitting the features values. This technique also provides in output an intuitive tree diagram and so it is also useful to understand the internal data structure [5].

Bayesian Classifier [24]: it classifies a record estimating the probability that the sample belongs to each class and choosing the most probable class. However, in a dataset every sample is almost unique, and so it's impossible to directly estimate the needed probability. For this reason the Bayes theorem is used. This theorem provides the method to estimate the posterior probability $P(Y|\boldsymbol{X})$ (probability of a class conditioned to the sample's value) in terms of the prior probability $P(Y)$, the class conditional probability $P(\boldsymbol{X}|Y)$ and the evidence $P(\boldsymbol{X})$:

$$P(Y|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|Y) \times P(Y)}{P(\boldsymbol{X})} \qquad (2.1)$$

Artificial Neural Networks: it's a network that can 'learn' a model of classification, changing its configuration. It is very used in machine learning. Analogous to human brain structure, an Artificial Neural Network is composed of an interconnected assembly of nodes and direct links [23].

Support Vector Machine: Support vector machine (SVM) is a technique of automatic learning for the classification of data. It is based on the idea of dividing the features space using a function that allows to separate the different classes [25].

2.2 Support Vector Machine

Support vector machine (SVM) is a technique of automatic learning for the classification of data. The idea is to divide the features space using a function that allows to separate the different classes.

Support Vector Machine is one of the most used techniques of classification because of its good performance in terms of accuracy and time consumption. This technique has shown promising empirical results in many practical applications, from handwritten digit recognition to text categorization. SVM also works very well with high-dimensional data and avoids the curse of

dimensionality problem. Another unique aspect of this approach is that it represents the decision boundary using a subset of the training samples, known as the support vectors.

There exists different kind of SVM. Considering linearity, SVM can be divided in linear and non-linear. Non-linear SVM is reduced to linear case using a kernel function that allows to create a space with more dimensions than the number of the features, so the mathematical techniques of linear case can be applied to the extended space.

Moreover, considering data distribution, linear SVM can be divided in two cases: linearly separable and not linearly separable. In linearly separable case there exists a boundary that allows to divide the feature space among different classes, while this is not possible in not separable case, where is needed to use a soft margin approach (where the decision boundary is built considering the possibility of misclassifying records in training phase). This approach is useful also in separable case, in order to avoid overfitting.

In the next we analyze the case of two classes. Then, when classes are more than two, there are algorithms that allow to reduce the problem to the two classes case. In the experiments of chapter 4, the *one versus all* approach is used.

2.2.1 Linear SVM: separable case

The base case of SVM is linear SVM in a separable problem, where the SVM is the optimal boundary that maximizes the distance between the groups of records of the same class. The goal is to found a function that describes the decision boundary in the space of features. In this case such boundary exists for hypothesis, so is used a hard margin approach that searches the optimal hyperplane dividing the dataset.

Firstly, a margin is defined as shown in Figure 2.1; the margin represents the distance from boundary to the nearest element of each class. Then, it's formalized an optimization problem on the margin length, which will be solved using Lagrange multiplier method.

Consider a binary classification problem consisting of N training samples. Each sample is denoted by a tuple (x_i, y_i) where $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,d})$ corresponds to the attribute set for the $i-th$ sample. Using conventionally $\{-1,1\}$ as classes, the decision boundary will have the form:

$$W \cdot x + b = 0 \qquad (2.2)$$

Where w and b are the parameters which have to be determined solving the optimization problem.

The first step consists in determine the coefficients W and b of (2.2). For every record, the following conditions must be verified:

$$W \cdot x_i + b \geq 1 \quad if\ y_i = 1 \qquad (2.3)$$

$$W \cdot x_i + b \leq -1 \quad if\ y_i = -1$$

These conditions can be summarized in the following:

$$y_i(w \cdot x_i + b) \geq 1 \qquad i = 1,2, \dots, N \qquad (2.4)$$

As in Figure 2.1, the margin in separable case is defined as the minimum distance between elements of different classes (in the figure it is represented the case of a space of two features, but it is the analog of the case with more features). The nearest elements to the boundary are called support vectors of the relative class. Considering two points of different class x_1 and x_2, the distance between the hyperplanes passing on the support vectors is determined as follow. Initially the parameters W and b are scaled in order to obtain hyperplanes passing on the support vectors described by the following equations:

$$W \cdot x_1 + b = 1 \qquad (2.5)$$

$$W \cdot x_2 + b = -1$$

So

$$W \cdot (x_1 - x_2) = 2 \qquad (2.6)$$

$$\|W\| \cdot d = 2$$

$$d = \frac{2}{\|w\|}$$

Where d is the distance between hyperplanes passing on the support vectors.

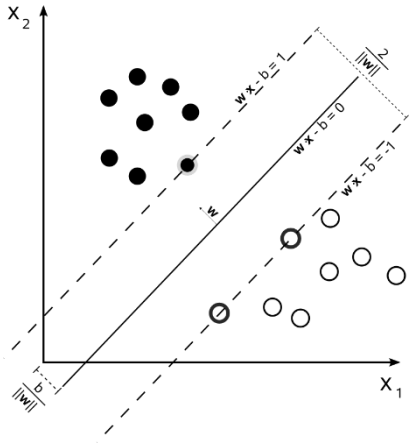


Figure 2.1: decision boundary for SVM in linear separable case.

To solve the problem of optimization of the separation, we need to maximize the margin, which means to minimize the function:

$$F(x) = \frac{\|w\|^2}{2} \qquad (2.7)$$

Considering the boundaries (2.3). We get the following optimization problem:

$$Objective\ function: \quad \min_{w} \frac{\|w\|^2}{2} \qquad (2.8)$$

$$Constraints: \quad y_i(w \cdot x_i + b) \geq 1 \quad i = 1,2, \dots N$$

This problem is a convex optimization problem, because the objective function is quadratic. It can be solved with Lagrange multipliers method.

Intuitively the process to use the Lagrange multipliers is the following:

1) We construct a Lagrangian function, which takes into account the boundaries:

$$L_p = \frac{1}{2}\|W\|^2 - \sum_{i=1}^{N} \lambda_i \cdot (y_i(w \cdot x_i + b) - 1) \qquad (2.9)$$

   Where λ_i are the Lagrange multipliers and must be determined

2) We can hypnotize $\lambda_i \geq 0$, in fact every solution with negative Lagrange multipliers can only increase the objective function

3) We impose the derivatives of L_P respect to w and b equal to zero:

$$\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{N} \lambda_i \cdot y_i \cdot x_i \qquad (2.10)$$

$$\frac{\partial L_p}{\partial b} = 0 \Rightarrow w = \sum_{i=1}^{N} \lambda_i \cdot y_i = 0$$

4) The set of inequality in the optimization problem can be replaced by equality, thanks to the fact that the Lagrange multiplier are greater than zero. These conditions are also called the Karush-Kun-Tucker conditions (KKT):

$$\lambda_i \geq 0 \qquad (2.11)$$

$$\lambda_i(y_i(W \cdot x_i + b) - 1) = 0$$

5) The problem is simplified considering the dual problem. The Lagrangian is transformed into a function of the Lagrange multipliers only.

When the problem will be solved, only the support vectors will have a value of $\lambda_i \neq 0$ and that will be used also to determine decision value and probability estimation in practical implementation.

2.2.2 Linear SVM: not separable case

In not separable case we can't define an optimization problem with a boundary that is valid for all records in the features space. So it's used a soft margin approach. Soft margin approach can be applied also to avoid overfittng, in the case where a classifier that has a low training error can generate low final accuracy. Soft margin approach is represented in Figure 2.2, where boundary B1 will provide a better description of the class distribution respect to boundary B2, also if B2 perfectly divides the dataset. We need to consider soft margin in constraints inequality (2.3) in this problem. This can be done introducing slack variables $\xi \geq 0$:

$$W \cdot x_i + b \geq 1 - \xi \quad if\ y_i = 1 \tag{2.12}$$

$$W \cdot x_i + b \leq 1 + \xi \quad if\ y_i = -1$$

$$\xi_i \geq 0$$

This allows to consider a wide margin, which can case classification errors in some case. To avoid this we need to modify the object function to the following:

$$f(w) = \frac{\|W\|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right)^k \tag{2.13}$$

Where C and k represent the cost parameters. For simplicity k is set to 1, and C will be chosen for example as better experimental choice in validation phase. If C is set as infinity or a high value, the situation will be the hard margin case.

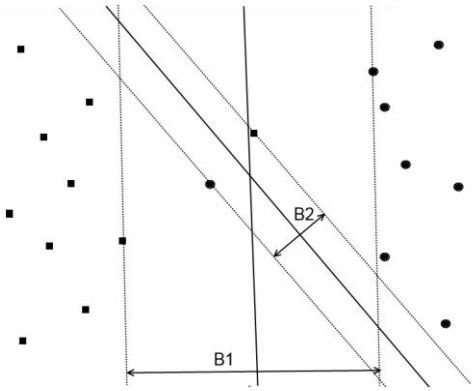


Figure 2.2: soft margin approach.

2.2.3 Non-linear SVM

Non-linear SVM takes into account the case in which the boundary function used to divide the feature space is not linear. This can generate a better accuracy, but is more computationally expensive than the linear SVM. To solve the problem of determine a non-linear boundary, the data are transformed from the original coordinate space in X into a new space $\phi(x)$, so that a linear boundary can be used to separate the instances in the new space. After the transformation of the space of features, the previous strategy can be apply to define a classification model.

After the transformation, the linear problem will be the following:

$$Objective\ function: \quad \min_{W} \frac{\|W\|^2}{2} \tag{2.14}$$

$$Constraints: \quad y_i(W \cdot \phi(x) + b) \geq 1 \qquad i = 1,2, \dots N$$

Which is similar to problem (2.8) except for the substitution of x by $\phi(x)$.

The dual Lagrangian problem has the following form:

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \ \varphi(\mathrm{x_i}) \varphi(\mathrm{x_j}) \tag{2.15}$$

And finally an element z can be classified using the function:

$$f(z) = sign(w \cdot \ \varphi(\mathrm{x}) + \mathrm{b}) = \mathrm{sign}\left(\sum_{i=1}^{n} \lambda_i y_i \ \varphi(\mathrm{x_i}) \varphi(\mathrm{z}) + \mathrm{b} \right) \tag{2.16}$$

This procedure has a high computational cost, because the multiplicative term $\phi(x_i) \cdot \phi(x_j)$ introduces non-linear terms. A strategy named Kernel Trick is adopted to simplify the non-linear problem.

Kernel trick is based on the idea that a function can be used to approximate the dot product in the new feature space with a product in the original feature space. A Kernel function has the following form

$$K(u, v) = \varphi(\mathrm{u}) * \varphi(\mathrm{v}) \qquad (2.17)$$

and commonly used functions are:

1) Polynomial: $K(x, y) = (x \cdot y + 1)^p$
2) Radial Basis Function: $K(x, y) = e^{(\|x-y\|^2\gamma)}$
3) Hyperbolic tangent: $K(x, y) = \tanh(\gamma \cdot x \cdot y + \delta)$

It's very important the right choice of the parameters, since this setting can influence the final performance. Usually these parameters are set in a validation phase. In this phase the model built on training set is tested on a validation set with different values of parameters and then the better parameter setting will be used in final experiments.

2.2.4 libSVM

libSVM is the MATLAB tool used in the implementation of the classification algorithms. This tool provides an implementation of SVM which offer also other options such as kernel choice or cost parameter settings.

LibSVM provide multiclass classification. However, in this thesis work, multiclass problems will be reduced to two class problem and then the original multiclass problem is reconstructed with *one-vs–all* approach. LibSVM solve multiclass problem using *one-vs-one* approach, which is more computationally expensive than *one-vs-all*, so we for simplicity adopt *one-vs-all*, reducing each dataset to the two classes' case.

With LibSVM it's possible to set the following parameters to obtain the previously described functions of SVM:

- -s parameter: it defines the type of SVM:

  0 -- C-SVC

  1 -- nu-SVC

  2 -- one-class SVM

  3 -- epsilon-SVR

  4 -- nu-SVR

- -t parameter: define kernel function:

  0 -- linear: u'*v

  1 -- polynomial: (gamma*u'*v + coef0)^degree

  2 -- radial basis function: exp(-gamma*|u-v|^2)

  3 -- sigmoid: tanh(gamma*u'*v + coef0)

We will use the default C-SVM with a cost parameter equal for both the classes. Cost factor is imposed trough the "–c" option.

Normally will be used a linear kernel (default), but in some cases, as when it is needed to use different classifiers (diversity based approach), other kernel types are used. Kernel parameters are set trough "–d", "–g" and "–r" options.

With reference to c parameter, in the following will be briefly introduced C-Support Vector Classifier. Given training vectors $x_i \in R^n \; i = 1, \dots, l$ and an indicator vector $y \in R^l$ such that $y_i \in \{1, -1\}$, C-SVC [14] solves the following primal optimization problem:

$$\min_{w,b,\xi} \; \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \tag{2.18}$$

$with\ constraints$: $y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i,$

$\xi_i \geq 0, i = 1, \dots, l$

Where $\phi(x_i)$ maps x_i into a higher dimensional space and $C > 0$ is the regularization parameter: a cost parameter which intuitively determines the costs of errors in clustering.

- c parameter will be often used in SVM classifiers to achieve better performance.

2.2.5 *One-vs-all* approach

For multiclass problem, the *one-vs –all* approach has been adopted. The multiclass learning problem is converted in binary class problems and then every sample is classified in each subproblem. Finally a definitive classification step assigns to each sample the class that has been more "strongly" assigned previously, which means "with highest decision value". For example, in a Support Vector Machine implementation, the considered decision value can be reasonably the distance from the boundary to the sample in the features space. More precisely, the problem can be mathematically described as follow.

Consider labels y, where $y_i \in \{1, \dots, K\}$ is the label of the sample X_i. The *one-vs-all* approach implements the following procedure.

For each class k in $\{1, \dots, K\}$, construct a new label vector $\tilde{y}$ where $\tilde{y}_i = 0$ when $y_i = k$ and $\tilde{y}_i = 1$ otherwise. The learning algorithm is applied to the new dataset $\{X, \tilde{y_k}\}$ for each class $k = 1, \dots, K$. In each iteration a binary problem is considered, and the score f_k determined by the classifier, represent the "weight" of the assignment of the class k to the sample. Then the class definitively assigned to the sample is determined as:

$$\hat{y} = arg \max_{k \in \{1,\dots,K\}} f_k(x) \qquad (2.19)$$

Performance evaluation

To determine if a machine learning model built on a training set is good or not, it is tested on a different set, and the values which determine the goodness of the model are calculated.

The accuracy is the ratio between the number of records correctly classified and the total number of records. However, it is not a parameter which represents completely the goodness of the used model. In many different situation it's necessary to consider the confusion matrix, for example in datasets with class imbalance problem. Moreover other measures can be used, such as ROC curve or contingency table. For the purpose of this thesis work only accuracy measure is reported in the results section.

2.3 Semi-Supervised Learning

Automatic learning can be divided in three types, considering the kind of data used in the learning process.

- **Supervised learning:** it is a learning process that uses only labeled data (data already classified, often manually and so for this type of learning usually are not available large training sets).
- **Unsupervised learning**: the learning process is applied to data which are not already classified. The most common case is clustering, which consists in grouping elements in different clusters. Examples of clustering are segmentation of customers based on similar buying patterns or identify similar web usage patterns.
- The third kind of learning is **Semi-Supervised Learning**, which takes advantage from the usage of unlabeled data to improve the performance of traditional supervised learning.

Semi-supervised learning have been developed successfully in the last years and provides good results in many practical applications. The study of semi-supervised learning is motivated by two factors: its practical value in building better computer algorithms, and its theoretical value in understanding learning in machines and humans.

Since in semi-supervised learning the training samples contains also unlabeled data, there are two distinct goal for machine learning:

- Predict the labels of future test data
- Predict labels of unlabeled instances of training samples

These problems are called *inductive semi-supervised learning* and *transductive learning* respectively [10].

Definition 2.3.1: Inductive semi-supervised learning: Given a training sample $\{\mathbf{x}\}$,$\{X\}$, inductive semi-supervised learning learns a function $f: X \to Y$ so that f is expected to be a good predictor on future data, beyond $\{\mathbf{x}\}$

Definition 2.3.2: Transductive learning: Given a training sample $\{\mathbf{x}\}$,$\{X\}$, tranductive learning trains a function $f: X \to Y$ so that f is expected to be a good predictor on the unlabeled data $\{\mathbf{x}\}$. Note that f is defined only on the given training sample, and is not required to make prediction outside. It is therefore a simpler function respect to the one defined in inductive semi-supervised learning.

There exists different model and approaches to semi-supervised learning and the usage of unlabeled data to improve performance of learning. A general classification of approaches that can be used divides them in generative models and discriminative models [2]:

- **Generative models:** Generative algorithms try to model the class-conditional density $P(x|y)$ by some unsupervised learning procedure. Applying the Bayes theorem we obtain

$$P(y|x) = \frac{P(x|y) * P(y)}{\sum_y P(x|y) * P(y)} \quad (2.20)$$

Which is an useful instrument to determine $P(x|y)$ when each record is almost unique (which is in practice every real case).

- **Discriminative models:** Discriminative algorithms do not try to estimate how the x_i have been generated, but instead concentrate on estimating $P(y|x)$. Some discriminative methods even limit themselves to modeling whether $P(y|x)$ is greater than or less than 0.5; an example of this is the semi-supervised version of support vector machine. It has been argued that discriminative models are more directly aligned with the goal of supervised learning and therefore tend to be more efficient in practice.

In the next, some models and techniques of semi-supervised learning and some specific algorithms will be introduced. Then in Chapter 3 the papers and algorithms used for the thesis work will be discussed.

When we work with real datasets, a very important limitation of supervised learning is the small size of training data. In many cases, labeled data represent records which have been manually analyzed and labeled, and the amount of this kind of data is often limited.

For example, for medical auto-diagnosis applications the labeled data are related to cases of patients which have been examined by a specialist and a diagnosis has been provided. The amount of this kind of data is limited for training process.

As can be seen simply applying a Support Vector Machine to the small set of labeled data, the accuracy is sensibly less than when the training set contains more labeled data. Following the protocol used in this thesis, the labeled data are 10% of training set, which is 80% of total data (the protocol will be illustrated in the next chapters).

However in real cases the only labeled data available are the data manually labeled, and for this reason they represent a relatively small quantity. The idea of semi-supervised learning is to use also unlabeled data, available from training set, to construct a better model of classification. Experimentally has been seen that this technique can provide better performance in data classification. In the next, some general assumptions about semi-supervised learning and its main drawbacks will be explained and then some base techniques (self-training, co-training and help training) and the algorithms used in the reference articles will be introduced.

2.3.1 Assumption for Semi-Supervised Learning

In many cases the semi-supervised learning techniques comport advantages respect to supervised learning. However there is an important prerequisite: the distribution of samples (labeled and unlabeled) have to be relevant for the classification problem. In other words, we could say that the knowledge of $p(x)$ that one gains through the unlabeled data has to carry information that is useful in the inference of $p(y|x)$. If this is not the case, semi-supervised learning will not yield an improvement over supervised learning. It might even happen that the usage of the unlabeled data degrades the prediction accuracy by misguiding the inference [2].

- **Semi-supervised smoothness assumption:** if two points x_1 and x_2 in a high density region are closed, then so should be the corresponding outputs y_1 and y_2.

This Assumption follow from the smoothness assumption which say that two points close each other in the space of features should have the same label. Intuitively this is important to make possible a generalization from a finite training set to a set of possibly infinitely many unseen test cases.

For example, if many points in the same region of features space have different labels, it's difficult for a classifier (for example a SVM) to determine a boundary dividing the classes. Moreover when

this boundary can be found, it's not very significant in test phase when test samples, for hypothesis, can be arbitrarily of one of the classes (because they are in a high density region).

- **Cluster Assumption:** if points are in the same cluster, they are likely to be of the same class.

This assumption is based on the idea of use unlabeled data to better define the clusters of classes. In this hypothesis the labeled data can be divided in clusters and unlabeled data can be used to find the boundary of each class more accurately.

The cluster assumption can be easily seen as a special case of the semi-supervised smoothness assumption, considering that clusters are frequently defined as being sets of points that can be connected by short curves which traverse only high-density regions.

The cluster assumption can be formulated as follow:

Low density separation: The decision boundary should lie in a low-density region.

Intuitively we can see that a decision boundary in a high density region would cut a cluster into two different classes. The presence of many objects of different classes in the same cluster would require the decision boundary to cut the cluster and then the boundary will not lie in a low density region.

- **Manifold assumption:** this assumption is about dimension of samples. Sometimes dimensions describing samples are too much (high-dimensional), so it's useful to reduce the number of dimensions and represent the problem in a different (low-dimensional) space (manifold). The manifold assumption can be summarized as follow:

Manifold assumption: the (high-dimensional) data lie (roughly) on a low-dimensional manifold.

2.3.2 Semi-supervised algorithms

Semi-supervised learning strategies can be divided in classes of algorithms, considering the used approach. For example four classes of algorithms can be defined, based on the following approaches [2]:

- **Generative models:** under the name *generative models* we refer to architectures following the generative paradigm described above. However quite all SSL algorithms are involved in the estimation of P(x|y) and from that derives the class probability estimation, instead of calculate it directly. So are classified as Generative models only technics strongly oriented to the determination of $P(x)$.

- **Low-Density separation**: this class contains algorithms which try to directly implement the low-density separation assumption by pushing the decision boundary away from the unlabeled points. The most common approach to achieving this goal is to use a maximum margin algorithm such a Support Vector Machine considering also unlabeled data (for example transductive SVM).

- **Graph-Based Method:** During last years, graph-based methods has been a very active area of research in semi-supervised learning. The common denominator of these methods is that the data are represented by the nodes of a graph, the edges of which are labeled with the pairwise distances of the incident nodes (and a missing edge corresponds to infinite distance).

- **Change of Representation:** these algorithms are not intrinsically semi-supervised, but instead perform a two-step learning.
    1. Perform an unsupervised step on all data, labeled and unlabeled, but ignoring the available labels. This can for instance, be a change of representation, or the construction of a new metric or a new kernel.
    2. Ignore the unlabeled data and perform supervised learning using the new distances, representation or kernel.

  This can be seen as direct implementation of the semi-supervised smoothness assumption, since the representation is changed in such a way that small distances in high-density regions are conserved.

Considering the different approaches, a Semi-Supervised Learning algorithm can combine some of them, or use other techniques like clustering. In the following, some popular SSL algorithms and basic approach to the SSL problem will be introduced.

2.3.3 Self-training

Self-Training is based on the idea of classifying progressively unlabeled data, adding to the pool of labeled data the records which have, at each step, a high ‘confidence’ in classification [8] [10].
In Self training a classifier is trained on an initial small amount of labeled data and is used to classify unlabeled data. For each classified element, is define a confidence measure which represent the ‘validity’ of assigned label. For SVM the confidence measure is usually the probability estimation, or the absolute value of the distance from the boundary. Both this values can be obtained in output of libSVM tool.

The most confident elements obtained from this step are added to the labeled data with the relative label resulting from classification, then the classifier is re-trained on new labeled data. The process is iterated until unlabeled data are less than a threshold parameter.

Self-training is characterized by the fact that the learning process uses its own predictions to teach itself. For this reason, it's also called self-teaching or bootstrapping (not to be confused with the statistical procedure with the same name). Self-training can be either inductive or transductive, depending on the nature of the predictor f.

Self-training

Input: labelled data $\{(\boldsymbol{x_i}, y_i)\}_{i=1}^{l}$, unlabeled data $\{\boldsymbol{x_j}\}_{j=l+1}^{l+u}$

1. Initially, let $L = \{(\boldsymbol{x}i, yi)\}_{i=1}^{l}$ and $U = \{\boldsymbol{x_j}\}_{j=l+1}^{j=l+u}$
2. Repeat
3. Train f from L using supervised learning
4. Apply f to the unlabeled data instances in U

Algorithm 2.1: Self-training pseudocode.

This semi-supervised learning method [10] is very simple to implement and has been applied successfully to several natural language processing task. In [26] self-training is used for word sense disambiguation, e.g. deciding whether the word 'plant' means a living organism or a factory in a given context. It has been used also to identify subjective nouns [27] and classify dialogues as 'emotional' or 'non-emotional' with a procedure involving two classifiers [28]. Self-training has also been applied to parsing and machine translation. In [29] self-training is applied to object detection systems from images, and shows how the semi-supervised technique compares favorably with a state-of-the-art detector.

Self-training is a very simply SSL algorithm to implement; is a wrapper method and it can be used in many complex algorithms. A disadvantage is, for example, that early classification errors can

‘propagate’ themselves and affect final accuracy, moreover there is a problem of convergence of the algorithm.

2.3.4 Co-training

Another popular algorithm for semi-supervised learning is co-training. This algorithm was originally proposed in [15]. In this technique two classifiers work together to perform label propagation. The classic approach of co-training is to train two classifiers separately based on two sufficient and redundant feature subsets (views), and then recover the most confident data for each other as the new labeled data. For this approach, there are strong assumptions on the feature sets.

Remark 2.3.1 Co-training Assumptions

- Features can be divided into two subsets and each subset is sufficient to learn a good classifier
- Two features subsets are conditionally independent given the class.

The first assumption on features independence is quite strong, since in a general dataset features has some kind of correlation between each other. In order to relax this assumption, variations of the algorithm have been proposed: a class of algorithms of co-learning, which doesn’t need a features separation for the two classifiers. Two learners, or an ensemble of learners, are trained separately on the full feature set of the labeled training data and then labels are predicted on the unlabeled data separately. In the case of an ensemble of learners a majority voting is used in order to determine the classification of unlabeled data.

Another variation of co-training is tri-training, proposed in [16]. In this algorithm three learners are used in the learning process. In detail, training data are generated by bootstrap sampling from initial data source. Then three hypotheses are trained based on these data. In the learning process, when two

learners agree with the classification of a new unlabeled sample, their prediction label will be marked on the unlabeled sample for the third learner. Three initial learners were updated during the tri-training process and the final hypothesis is determined by the majority voting.

Co-training model has practical applications in many real world cases. For example co-training model was applied for the named entity classification in the natural language process. Co-training with error-correcting output codes (ECOC) was applied for text classification. For visual detection, co-training has reduced the false positive rate significantly. It has also been successfully applied to email classification and statistical parsing [10].

2.3.5 Help training

Help-training is a semi-supervised algorithm proposed in [17]. The idea of this algorithm is to use a classifier based on a generative model to 'help' a classifier which uses discriminative approach.

Let consider the main classifier C which is based on a discriminative approach and the classifier G based on a generative model. The classifier G produces a probability density model. It tries to find the basic information of each class by modeling the data. In Help-Training, the classifier C is helped by G to make decisions about which samples can be labeled and added to the training set. So, at each iteration, the classifier G is used to select the samples which have a high probability to belong to each class. These selected samples constitute the candidate samples for labeling process. The classifier C classifies the pre-selected samples and those that are classified with highest scores are added to the training set. The process is repeated until all unlabeled data are labeled.

2.3.6 Adaptive bootstrap

Adaptive bootstrap is a supervised classification technique which train a group of different classifiers (usually decision tree) on different datasets constructed from the original dataset by bagging with replacement $n \leq N$ data, where N is the dimension of the original dataset. Then, from this initial group of classifiers, a single learner it is derived combining together the classifiers.

This technique provides good results and so it's quite diffused.

For the experiments of this work the MATLAB library adaboost is used, which follows the syntax:

- for the training the function fitensamble is used:

  `Ensemble = fitensemble(X,Y,Method,NLearn,Learners, Name,Value)`

  Input:

  `X:` data

  `Y:` labels

  `Method:` the method of classification: for classification with Ada Boost, this parameter is the string 'AdaBoostM1'

  `NLearn:` the number of ensemble learning cycles, i.e. the number of times the entire procedure will be repeated

  `Learners:` the basic kind of learner, typically decision tree

  `Pairs (Name, Value):` define other options, for example a cost matrix.

  Output:

  `Ensemble:` the classifier built form the ensemble of learners

- For prediction the function predict is used:

```
[predict_label,scores] = predict(Ensemble,data)
```

Input:

`Ensemble:` the ensemble of classifier obtained from fitensamble

`Data:` unlabeled data

Output:

`Predict_labels:` predictions of the labels for input data

`Scores:` a measure relative to probability estimation

2.4 Risks of Semi-Supervised Learning

Empirical and theoretical results have often testify favorably to the semi-supervised learning of generative classifiers. However, the literature has also brought to light a number of situations where semi-supervised learning fails to produce good generative classifiers.

For example, [18] reports experiments where unlabeled data degraded the performance of Naïve Bayes classifiers with Gaussian variables. The authors attribute such cases to deviations from modelling assumptions, such as outliers or "samples of unknown classes", they even suggest that unlabeled samples should be used with care, and only when the labeled data alone produce a poor classifier. Another representative example is [30], where classifiers sometimes display performance degradation. The authors suggest several possible sources of difficulties: numerical problems in the learning algorithm, mismatches between the natural cluster in feature space and the actual labels, etc. In [19], labeled and unlabeled data are used to learn Bayesian network classifiers, from naïve Bayes classifiers to fully connected networks. The naïve Bayes classifiers display bad classification performance, and in fact the performance degrades as more unlabeled data are used (more complex networks also display performance degradation as unlabeled samples are added).

A last example is provided in [20]. In the described experiments outliers are added to a Gaussian model, producing a performance degradation of generative classifiers.

2.4.1 Methodological Considerations

Given a pool of labeled and unlabeled data, generative semi-supervised learning is an attractive strategy. However, one should always start by learning a supervised classifier with the labeled data. This "baseline" classifier can then be compared to other semi-supervised classifiers through cross-validation or similar techniques. Whenever modelling assumption seem inaccurate, unlabeled data can be used to test modelling assumptions. If time and resources are available, a model search should be conducted, attempting to reach a "correct" model (that is a model where unlabeled data will be truly beneficial).

An additional step consists in the comparison of the baseline classifier to non-generative methods. There are many semi-supervised non-generative classifiers, and there are also a significant number of methods that uses labeled and unlabeled data for different purposes (for example methods where the unlabeled data are used only to conduct dimensionality reduction). However we should warn that a few empirical results in the literature suggest the possibility of performance degradation in non-generative semi-supervised learning paradigms, such as transductive support vector machine.

2.4.2 Active Learning

A final methodological comment concerns active learning. This technique consists in labelling selected samples among the unlabeled data, instead of use the entire set of unlabeled data. This option should be seriously considered whenever possible. In some cases has been observed that the most profitable usage of unlabeled data is to use these data as a pool of samples from which some samples

can be carefully selected and labeled. In general, we should take the score of a labeled sample to be considerably higher than the score of an unlabeled sample.

Chapter 3

Proposed system

This thesis is based on the implementation and combination of four state-of-art techniques described in the relative articles of reference. Then the techniques are combined together through specific protocols and the experiments result are shown.

In this chapter, the used algorithms will be explained. So the algorithms and articles are presented by a theoretical point of view in this chapter, while in the next chapter, tests and experiments will be proposed.

3.1 Reference Articles

3.1.1 DCPE co-training for classification

In [7] a method for semi-supervised classification is proposed. The idea is to use diversity of class probability estimation (DCPE) of two different classifiers to add progressively to labeled data the unlabeled data with high DCPE. This algorithm is based on the idea of co-training, which uses two different classifiers to evaluate ‘better’ unlabeled data which can be added to labeled data with a label predicted with high confidence.

To implement this approach two different supervised algorithms have been used: a support vector machine with polynomial kernel, and an implementation of adaptive bootstrap provided by MATLAB. The pseudocode of the algorithm is the described in Algorithm 3.1:

Pseudocode describing the DCPE co-training algorithm

1: Input:
   b1 baselearningalgorithm1
   b2 baselearningalgorithm2
   L labeled data
   U unlabeled data
2: Allocate $L_A = L$, $L_B = L$
3: Use b1 to train on L_A to get a classifier h_A, uses b2 to train on L_B to get a classifier h_B
4: Create a pool U' by randomly choosing u data from U
5: if size(U) $\geq$ u then
6: Use h_A and h_B separately to predict each data x in U': $h_A(x)$, $h_B(x)$. Record class probability estimation (cpe) for each data: $cpe_A(x)$, $cpe_B(x)$
7: Update $L_A = L_A + l_a$, l_a are chose from U', which have same prediction labels ($h_A(x) = h_B(x)$) and the highest class probability estimation differences between h_B and h_A:
$$x = arg\max_x(cpeB(x) - cpeA(x))$$
8: Update $L_B = L_B + l_b$, l_b are chose from U', which have same prediction labels ($h_A(x) = h_B(x)$) and the highest class probability estimation differences between h_A and h_B:
$$x = arg\max_x(cpeA(x) - cpeB(x))$$
9: Remove la, lb from U'
10: Randomly choosing new u data from U to replenish U'
11: Update h_A by using b1 to train on L_A, update h_B by using b2 to train on L_B
12: end if

Algorithm 3.1: DCPE Co-training algorithm.

This algorithm implements a scheme of co-training, based on the diversity of class probability estimation. In ensemble learning, diversity plays an important role in combining different learners. There exists many different approaches to use diversity in learning process. Some of the most diffused are:

- Diversity based on **features subsets**: features are partitioned in two subsets and then a base classification algorithm is used in each subset to determine a measure of diversity. Then co-training algorithm is applied. This approach is also called *two view co-training*.
- **Random features splitting**: instead of divide features in subsets, features are randomly selected. This approach was performed in email classification and provided results comparable or better than using the original and natural features partitions.
- Use **different base learning algorithms**: this method establish two classifiers by applying two base learning algorithms on the whole features set. This kind of co-training is called *single view co-training*, since diversity is calculated on the entire features set.

The implemented algorithm uses the third approach, which presents several advantages. It doesn't require a feature selection and so it's simpler and easy to implement. Moreover it doesn't require the co-training assumptions to be satisfied; this means that is not necessary that each view of the dataset represents exhaustively the data structure, and the two view are not necessarily conditionally independent. This situation is very common in real cases, so this approach can be considered more robust for general practical applications.

algorithm description

The DCPE co-training algorithm is designed for binary classification, and can be converted to multiclass problem using a technique such as one-vs-all approach.

As shown in the block diagram of Figure 3.1, from unlabeled data U are selected u samples (in the experiments is used $u = 20$) to form a pool of unlabeled data U'. It's important to note that the selected data are removed from U after selection.

Two labeled data sets are defined: L_A and L_B. Initially they are sets equal at the entire labeled data set $L_A = L_B = L$. In successive iterations L_A will be updated with unlabeled data with high class probability estimation difference from h_B classification to h_A classification and vice versa for L_B.

Then two different classifiers h_A and h_B are trained on the labeled data and are used to classify data in U'. Data classified whit high difference in *cpe* (class probability estimation) from h_A to h_B are added to L_A and with high difference in *cpe* from h_B to h_A are added to L_B. Selected data with high *cpe* are removed from $U'$and Up is refill by sampling new u data from U and then the classifiers h_A and h_B are retrained respectively on the new labeled data set L_A and L_B. The process is repeated since there exist more than u unlabeled data in U.

At the end of each iteration, a classifier is trained on the total labeled data (union set of L_A and L_B). This classifier is tested on validation set and its accuracy is stored. At the end of the algorithm execution, a graph of the performance of each classifier is constructed, varying the number of iterations. On this graph the best classifier is chosen using a sliding window of dimension 3, as described in the reference article [7]. So the validation phase is implemented considering the number of iterations of the algorithm as the parameter to optimize.

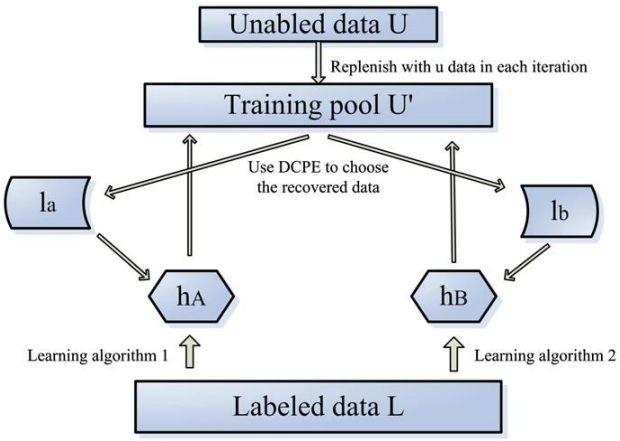


Figure 3.1 DCPE Co-training block diagram.

analysis and conclusions

There exist many different basic machine learning algorithms that provide good performance in different kind of dataset and learning problems.

Most popular learning algorithms are such as Naive Bayes, neural networks, k-nearest neighbor, decision tree, and others. An advantage of using a co-training technique is that it can compare two different algorithms, and eventually can reveal performance differences in the usage of different pairs of algorithms for a specific dataset or learning problem. So it can be useful to investigate the ensemble learning approach with algorithms that approach data classification from different perspectives.

Moreover, since diversity plays a critical role in ensemble learning methodology, the DCPE Co-training approach can also have an essential impact on the co-training based algorithms.

The major advantages and positive aspects of DCPE Co-training approach can be summarized as follow:

- DCPE co-training does not need the sufficient and redundant views (features subsets) of data set. The diversity of different learning algorithms is used to do label propagation. This can simplify the model and the implementation and, as said above, the single view co-training doesn't require the satisfaction of the Co-training Assumptions (Remark 2.3.1).
- A theoretical analysis can show how DCPE Co-training can achieve higher classification accuracy in co-training process. It can be proved that

  Theorem 3.1.1*: In the process of co-training, when the recovered data (with predicted labels) in i-th iteration have relatively smaller noise compared with labeled data, the learning hypothesis can achieve lower error.*
- Competitive results among classical supervised learning methods and semi-supervised learning method (co-training, self-training and tri-training) are obtained on binary UCI data sets.
- In the co-training process, the disagreement between two classifiers can be used for improving the final hypothesis. In DCPE Co-training method, is established the diversity between two classifiers, which shares similar property with disagreement-based co-training (a co-training approach proposed in [21]).

The key idea of this approach is to use the classification diversity from different learners for label propagation. It is a kind of ensemble learning approach based on the diversity of classifiers. The experimental results provided in [7] demonstrates that the proposed approach can achieve competitive results when it is compared with supervised learning methods and semi-supervised learning methods (co-training, self-training, and tri-training).

3.1.2 Using clustering analysis to improve semi-supervised classification

In the reference article [9] is proposed an algorithm which uses semi-supervised clustering combined with a traditional supervised SVM to classify data.

Clustering is, in its simplest interpretation, an unsupervised learning technique. It is the classical unsupervised problem. It consists in the division of samples in distinct clusters, considering their distribution in the space of features. Typically these data are not labeled, for example data can be relative to webpages which need to be grouped by similar argument. In this case the possible arguments of pages are not known a priori, or the argument itself is not important for the learning purpose. Moreover there could be webpages not classified of a specific argument, and they must be grouped for similar topic of discussion.

The corresponding version of semi-supervised clustering is based on the fact that there are also labeled data in the dataset, and so these data must be considered in the clustering process. In the algorithm proposed in [9], the labeled and unlabeled data are firstly elaborate in a semi-supervised learning way through semi-supervised clustering, labels are assigned to unlabeled samples and unlabeled samples classified with high confidence are take into account. Then labeled data obtained

at the previous step are classified with a supervised SVM. Then the process is iterated updating the sets of labeled and unlabeled data.

In particular the ideas underling this approach of analysis are the following:

- As unlabeled data may contain crucial information about the data space, clustering methods is used to reveal the underlying data space structure and to improve the training efficiency of the classifier.
- Labeled data are used to guide the clustering process through semi-supervised clustering methods.
- Newly labeled data are used not only to update the classifier (as in Self-training), but also to better guide the semi-supervised clustering methods.

The algorithm used to implement clustering is Semi-Supervised Fuzzy C-Mean (SSFCM), the semi-supervised version of Fuzzy C-Mean (FCM), a popular clustering algorithm.

semi-supervised fuzzy c-mean

As said before, SSFCM is the semi-supervised version of FCM. Fuzzy C-Means (FCM) is one of the most popular unsupervised clustering methods. In comparison to hard clustering, FCM provides an additional conceptual enhancement by allowing a data point to be assigned to different classes with various membership degrees. In this way, the patterns can be treated in a more reasonable way and the algorithm is capable of identifying eventual ‘‘outliers’’.

Let $X = \{x_1, x_2, \dots, x_n\}, x_i \epsilon \mathbb{R}^d$ be a dataset of size n and dimension d. Let’s consider a problem of clustering where c is the number of classes and V represents the set of prototypes associated with classes (prototypes are labeled samples chosen for each class).

The fuzzy c-mean solution of the problem consists in determining the partition matrix U which minimize the following objective function:

$$\min \Im_m = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m \parallel x_j - v_i \parallel^2 \qquad (3.1)$$

The superscript m is the degree of fuzziness associated with the partition matrix. U is a partition matrix whose element $u_{i,j}$ indicates the membership degree of the data point x_j to class i and satisfies two conditions:

$$0 \leq u_{ij} \leq 1 \qquad (3.2)$$

$$\sum_{i=1}^{c} u_{ij} = 1$$

Finally $d_{ij} = \parallel x_j - v_i \parallel^2$ expresses the distance between x_j and v_i.

Semi-Supervised version of FCM clustering is based on the idea of use labeled data to improve the performance of clustering. Labeled data can be used for choosing the prototype v_i and to provide additional information to the learning problem. Considering labeled data, the objective function to minimize assumes the following form:

$$\min \Im_m = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m \parallel x_j - v_i \parallel^2 + \sum_{i=1}^{c} \sum_{j=1}^{n'} u'^m_{ij} \parallel x'_j - v_i \parallel^2 \qquad (3.3)$$

Where the second term take into account the labeled data. Labeled data are $X' = \{x'_1, x'_2, \dots, x'_n\}$ where $x'_i \epsilon \mathbb{R}^d$. The partition matrix U' defines the membership degree of each predicted value to each class $1, \dots, c$ and must satisfy the following constraints

$$0 \leq u_{ij} \leq 1 \qquad (3.4)$$

$$\sum_{i=1}^{c} u_{ij} = 1$$

$$u'_{ij} \geq u'_{kj}, \quad \forall k \in \{1,2,\dots,c\}/\{i\}, \quad j = 1,2,\dots,n' \quad if\ L(x'_j) = i$$

Respect to the supervised case, an additional constraint is added. This constraints means that, for labe1ed data, the membership degree of the effective class i must be higher than the membership degree of each other class $j = 1 \dots c, j \neq i$.

Applying the Lagrange multiplier method we obtain the following equations for the optimal solution:

For data prototypes:

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j + \sum_{j=1}^{n'} u'^m_{ij} x'_j}{\sum_{j=1}^{n} u_{ij}^m + \sum_{j=1}^{n'} u'^m_{ij}} \tag{3.5}$$

For unlabeled data point x_j:

$$u_{ij} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d_{ij}}{d_{lj}}\right)^{2/(m-1)}} \tag{3.6}$$

For labeled data point x'_j

$$u'_{ij} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d'_{ij}}{d'_{lj}}\right)^{2/(m-1)}} \qquad if\ L(x'_j) = i \tag{3.7}$$

$$\min\left\{u'_{ij}, \frac{1-u'_{ij}}{\sum_{l=1, l\neq i}^{c} \left(\frac{d'_{kj}}{d'_{lj}}\right)^{2/(m-1)}}\right\}, \forall k \in \{1,2,\dots,c\}/\{\mathrm{i}\},\ \text{if } L(x'_j) = i$$

proposed algorithm

The algorithm proposed in [9] is essentially a framework for semi-supervised classification where a semi-supervised clustering process (SSFCM) is integrated into Self-training.

In the first phase, semi-supervised clustering uses both labeled and unlabeled data in the learning phase and assigns a label to unlabeled data, with the correspondent confidence degree (a measure of class probability estimation). Unlabeled data classified with higher level of confidence are selected for the supervised learning phase, while others values are left in the unlabeled set.

In the second phase, a supervised SVM is trained on labeled data and high-confidence classified unlabeled data with relative labels (provided by the first phase). The SVM is tested on the same

dataset used for learning, and data classified with high confidence are definitely added to labeled set, while other data are discarded and reinserted in the unlabeled set.

The two phase process is repeated until the unlabeled data are less than a parametric threshold. A diagram of the algorithm is represented in Figure 3.2. Then in the following the algorithm is presented in a more detailed way.

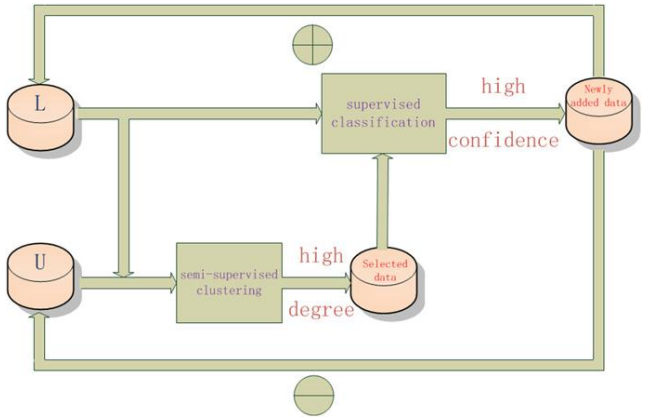


Figure 3.2 block diagram for Clustering and SVM algorithm.

SSFCM + SVM Pseudocode

Input: labeled dataset $L^{(0)}$, unlabeled dataset $U^{(0)}$

Output: SVM classifier

Method:

1. Initialize the dataset $L=L^{(0)}$ and $U=U^{(0)}$, threshold values: ε_1, ε_2, N
2. Repeat until $|U| \leq N$
   – Estimate the membership degree using SSFCM for unlabeled data
   – Select a dataset T1 where each sample xj has high certainty of belonging to one class.
   – Train the SVM with L – Compute the output f(x) of the SVM for the selected dataset T1
   – Select a dataset T2 where the output of each sample x by the SVM has high values.
   – Update the current labeled set $L \leftarrow L \cup T_2$
   – Update the current unlabeled set $U \leftarrow U - T_2$
   – Reduce the value of ε_1 if $T_2=\emptyset$
3. Label the remaining unlabeled data with the trained SVM, if $U \neq \emptyset$
4. Retrain the SVM

Algorithm 3.2: Semi-supervised fuzzy c-means.

In this implementation is important to note the thresholds ε_1 and ε_2. They represent the threshold for confidence respectively in the clustering semi-supervised classification and in the SVM supervised phase. When T_2 is empty, the value of ε_1 is reduced of 0.05, as described in the reference article [9]. This guarantee the convergence of the algorithm (since there is a value of threshold ε_1 that allows to add enough element to labeled set) and the while loop will terminate.

3.1.3 A semi-supervised feature ranking method with ensemble learning

This article [8] presents a method to determine a ranking of features, based on their importance in learning algorithm.

The proposed framework is based on semi-supervised learning, more precisely it uses a simple but efficient and easy to implement Self-training approach.

Feature selection consists in selecting a subset of relevant features, which achieves a better performance in accuracy for many learning problems with a large number of features.

Feature selection techniques provide three main benefits when constructing predictive models:

- **Improved model interpretability**: in many cases there is a large number of features and many of them are irrelevant or almost irrelevant for the purpose of classification. However is not known which features are irrelevant and which not, so it may be useful a feature ranking algorithm
- **Shorter training times**: training time is linear or exponential correlated to the number of features (it depends on the learning algorithm); so, reducing the number of features, the learning time can be considerably decreased.

- **Reduced generalization error by reducing overfitting**: overfitting is a common problem in many datasets, especially datasets with a high number of features. These datasets contain redundant information that must be 'weighted' in some way to permit to learning algorithm to consider only important features.

Feature selection is also useful as part of the data analysis process, as it shows which features are important for prediction, and how these features are related.

As machine learning, also feature selection problem can be classified considering the kind of used data:

- **Supervised feature selection** algorithms rely on measures that take into account the class information. A well-known measure is *information gain*, which is widely used in both feature selection and decision tree induction
- **Unsupervised feature selection** is more difficult to deal with than supervised feature selection. However, it is a very useful tool when the majority of data are unlabeled.

  Dy and Brodley [22] define the goal of feature selection for unsupervised learning as:

  > *« To find the smallest feature subset that best uncovers "interesting natural" groupings (clusters) from data according to the chosen criterion. »*

  Unsupervised feature selection can provide good performance, since great quantity of data are unlabeled.
- **Semi-supervised t** is the approach that uses both labeled and unlabeled data and is the approach used in the algorithm proposed in [8]. Since there is a large number of unlabeled data and a small number of labeled instances, it is reasonable to use unlabeled data to form

some potential clusters and then employ labeled data to find those clusters that can achieve both locality-based and class-based separations.

proposed algorithm

The algorithm consists in a novel semi-supervised features ranking algorithm, termed as *semi-supervised ensemble learning guided feature ranking* (SEFR).

The algorithm ranks features through an ensemble framework, in which a feature relevance is evaluated by the relative accuracy evaluated using both labeled and unlabeled data. The algorithm consists principally of two strategies:

- A combination of both data resampling (bagging) and random subspace strategies to generate an ensemble of semi-supervised classifiers which permit the exploration of distinct views of inter-pattern relationships.
- An extension of the Random Forest permutation importance measure [32], using the labeled and unlabeled data together; it is proposed to measure feature relevance.

A ranking of all features is finally obtained with respect to their relevance in all obtained semi-supervised classifiers.

In details, the SEFR algorithm follows these steps:

- Select with replacement m samples from U to form U_{bag} and n from L to form L_{bag}. Remaining samples forms respectively U_{oob} and L_{oob} (out-of-bound samples).
- From features set F select randomly $\sqrt{|F|}$ features.
- Until U_{bag} is not empty or there are no 'high confidence' classified unlabeled data, select the samples with highest score classified by ϕ (the base learning algorithm received in input).

This process is similar to Self-training approach, in fact in each iteration high confident samples are added to L_{bag} and so they influence confidence threshold calculation (see *selectMostConfident* algorithm).

- Determine *L'* and *U'* as high confidence classified samples in L_{oob} and U_{oob}.
- For each feature f in *F'*, perform a random permutation of its values in *L'* and *U'* and classify "permuted" samples. If the classification label is different from the real (or high confidence classified) label, increment $imp(f)$. $imp(f)$ represent the inverse importance rank of feature f.

The process is repeated $nbags$ times to provide different views of data (L_{bag}, U_{bag}) and features (*F'*). L_{bag} and U_{bag} are drawn with replacement (bagging) from original labeled data set *L* and unlabeled dataset U. Moreover $nbags$ is chosen considering the following formula:

$$nbags = 5 * ceil\left(\frac{\log(\alpha)}{\log(1 - 1/\sqrt{p})}\right) \quad (3.8)$$

Where p is the number of features. This formula ensures that each feature is drawn at least five times at a confidence level of $\alpha = 0.01$. The pseudocode of the algorithm is represented in Algorithm 3.4.

confidence measure

An important factor that affects the performance of any semi-supervised algorithm is how to measure the confidence of the labeling of an unlabeled sample, which determines its probability of being selected.

In [8] the proposed confidence measure is a normalized value of the membership degree in output of the base classifier (for SVM implemented in libSVM, it corresponds to the *decision value*).

The score $s_i(\boldsymbol{x})$ of a sample $\boldsymbol{x}$ for a class i is normalized as follows:

$$\widehat{p_i}(x) = \frac{s_i(\boldsymbol{x})}{\sum_{i=1}^{c} s_i(\boldsymbol{x})} \tag{3.9}$$

The value $\widehat{p_i}(x)$ can be considered an estimation of the conditional probability $p(i|x)$. This confidence measure is based on the probabilistic interpretation of scores.

The process to determine confidence in semi-supervised learning involves both labeled and unlabeled data. Consider the learning dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{l+u}$ containing labeled data $\boldsymbol{x_i}$, $1 \leq i \leq l$ and unlabeled data $\boldsymbol{x_j}$, $l+1 \leq j \leq u$.

Firstly the labeled data are used to determine a base classifier and evaluate the confidence level of unlabeled data. The classifier is mathematically represented as $\phi: \mathcal{D} \longmapsto \{1,2,\dots,c\} \times [0,1]$ which associates to any element x in dataset $\mathcal{D}$ a label $y(x)$ and an estimation of probability $p(y(x)|x)$. The probability estimation represents the confidence measure. For labeled data probability estimation is define equal to 1. Moreover is defined the set of misclassified labeled data:

$$\mathcal{D}_{\neq} = \{\boldsymbol{x_i} | \, \widehat{y_i} \neq y_i \, , 1 \leq i \leq l\} \tag{3.10}$$

And for each label $1 \leq i \leq c$ is defined the dataset (subset of labeled data):

$$\mathcal{D}_i = \{\boldsymbol{x_j} | \, \widehat{y_j} = i \, , 1 \leq j \leq l\}. \tag{3.11}$$

The intersection of (3.10) and (3.11) is the set of misclassified data of class i. In order to evaluate most confidently classified labeled data, considering the classifier constructed using the base learning algorithm, the following quantity is define:

$$\widehat{p_i}^{\,*} = \max\{\widehat{p_i} | \boldsymbol{x_j} \in \mathcal{D}_{\neq} \cap \mathcal{D}_i \, , 1 \leq j \leq l\} \tag{3.13}$$

Where $\hat{p}_i^*$ is an estimation of the maximal value of the probabilities $p(i|\boldsymbol{x})$ among all misclassified data of class i: $\mathcal{D}_i \cap \mathcal{D}_{\neq}$. Quantity $\hat{p}_i^*$ is used as a tolerance threshold for the labeling strategy of unlabeled data: unlabeled record $\boldsymbol{x}_i$ is labeled with its predicted label y_i if the corresponding class probability estimation $\hat{p}_i$ is greater than $\hat{p}_i^*$. The pseudocode of the algorithm used to determine most confident unlabeled data is shown in Algorithm 3.3. A similar method is also used to find most confidently labeled data, using the same principle of the threshold $\hat{p}_i^*$.

SelectMostConfident - Pseudocode
Input: D, L, U, ϕ
Output: U'
Method:
($\hat{\mathbf{y}}$, $\hat{\mathbf{p}}$) = ϕ (D (L ∪ U));
$L^{\neq} = \{l \in L; \hat{y}_l \neq y_l\}$;
U' = ∅ ;
for c = 1 : nc **do**
Lc = { l ∈ L; yl = c};
$L_c^{\neq} = L^{\neq} \cap L_c$;
if $L_c^{\neq} \neq \phi$ **then**
τ = max$\hat{p}(L_c^{\neq})$;
end
else
τ = 0;
U> = {u ∈ U; $\hat{p}_u$ > τ };
if isempty(admissInds) **then**
Uc = { u ∈ U; $\hat{y}_u$ = c};
Uc> = U> ∩ Uc;
U' = U' ∪ Uc>;
end
end

Algorithm 3.3 Select Most Confident unlabeled data.

SEFR - Pseudoocode

Input: D, L, U, F= {f_1,…,fp}, *nbags*, *BaseLearn*, ϕ
Output: F, *imp*
Method:

```
Imp = 0;
for i = 1:nbags do
    L_bag= bootstrap sample from L;
    U_bag= bootstrap sample from U;
    L_oob = L\L_bag;        U_oob = U\U_bag;
    Randomly draw √|F| features from F to form F';

    /* Labeling by self-training */

    while Ubag ≠ ∅ do
        U' = selectMostConfident (D (L_bag ∪ U_bag; F'), L_bag, U_bag, φ);
        if U' = ∅ then
            break;
        end
        L_bag = L_bag ∪ U';
        U_bag = U_bag \ U';
    end

    /* Feature importance measures */

    Apply φ to D (L_oob ∪ U_oob, F');
    Select the well classified samples in L_oob to form L';
    U' = selectMostConfident (D (L_oob ∪ U_oob, F'), L_oob, U_oob, φ);
    Define y as the predicted labels of L' ∪ U';

    for each f ϵ F' do
        Randomly permute f in D to form permD;
        Apply φ to permD(L'∪ U', F');
        Define y_p as the predicted labels of L'∪ U';
        Increase imp(f) by the number of mismatches between y and y_p;
    end
end
Rank the features f according to imp(f) and return both F and imp;
```

Algorithm 3.4 semi-supervised feature ranking.

3.1.4 Single Classifier-based Multiple Classification Scheme

Normally in many problems of pattern recognition, different classifiers perform differently and provide different accuracy values. So it's useful to test different classification algorithms on the dataset and select the one with better performance.

However, it has recently become common practice to use more than one classifier rather than a single one for pattern recognition tasks. In order to have a pool of classifiers with different errors, it is advisable to create diverse classifiers. To achieve this, the classifiers are grouped together into what is known as an Ensemble of Classifiers (EoC), and the approach that uses multiple classifiers to enhance classification accuracy is known as a Multiple Classifier System (MCS).

ensemble of classifiers (EoC)

One of the issues that is critical to the success of an *EoC* routine is the need for diversity in ensemble creation. For an *EoC* to perform well, errors produced by every classifier must be relative to different data points; so, when the *EoC* outputs are combined, most of the errors committed by classifiers will cancel each other out, and so the overall *EoC* will achieve a more accurate recognition rate. This property is referred to as the *diversity of an ensemble*.

To generate multiple data subsets there exists several methods which generate diversity. Diffused methods are:

- **Bootstrap:** it consists in generate new points from the existing samples in dataset. These points are generated for example duplicating existing samples.
- **Random subspacing:** this method creates various classifiers using different subsets of features to train them. Because problems are represented in different subspaces, different classifiers develop different borders for the classification.

- **Bagging:** Bagging generates different classifiers by randomly selecting subsets of samples from training set. Intuitively, selecting different subsets of data, it's expected a "diversity" in the classifiers that will be generated, and consequently in the classification of test samples.

Many *EoC*s are based on a two steps strategy: firstly an ensemble of different classifiers is generated, and subsequently is determined an optimal subset of these classifiers. More in detail is used a process consisting in these phases:

- Multiple data subsets are generated
- Multiple classifiers are trained with corresponding data subsets
- Adequate classifiers need to be selected from the pool of trained classifiers

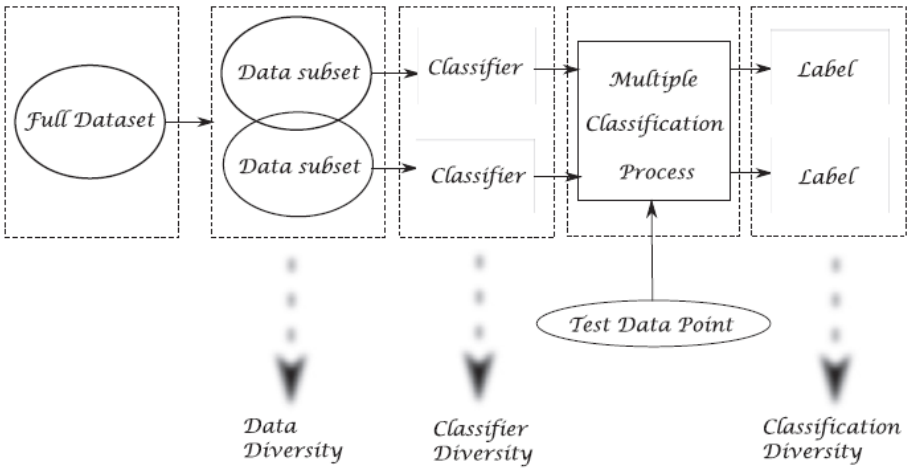
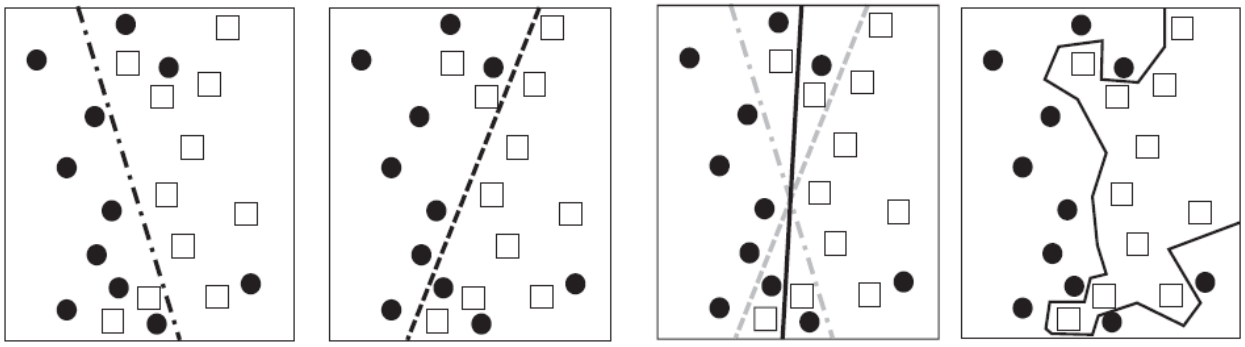


Figure 3.3 Three steps strategy in EoC approach.

The three-step process of overproduction and selection has, in fact, become a standard process in MCS, and its embedded complexity is accepted as an unavoidable cost. Moreover there are some problems related to the diversity measure. For example there is no a universal definition of diversity. Furthermore, it has been observed in many recent studies that clear correlations between ensemble

accuracy and diversity measures cannot be found in any of the existing diversity measure [33] [34] [35] [36].

An important step of an *EoC* based approach is the fusion of classifiers previously obtained. Considering that each classifier generates a boundary (for example consider a SVM as classifier), the fusion phase consists in the identification of a new boundary by applying a *fusion function*, that is a combination of the different boundaries drawn by different classifiers, as shown in Figure 3.4.



(a) Boundary drawn by classifier A

(b) Boundary drawn by classifier B

(c) Combined boundary obtained from fusion function

(d) True boundary (oracle)

Figure 3.4 Illustration of a boundary change by applying a classifier combination: dark circles represent samples from one class, and empty rectangles represent samples from another class; lines represent boundaries drawn by classifiers: (a) boundary drawn by classifier A; (b) boundary drawn by classifier B; and (c) new boundary created by combining the boundary from (a) and the boundary from (b), represented by a solid black line.

As can be seen in Figure 3.4, it can be difficult to achieve the oracle with a limited number of classifiers using a linear fusion function. Moreover, ensemble of classifiers approach presents other drawbacks and issues to be resolved, such as:

- The possibility of extracting diversity from a dataset directly without training multiple classifiers.
- Complexity reduction for the MCS, due to the omission of multiple classifier training and classifier selection.

- The feasibility of multiple classifications without multiple classifiers by using diversity.
- The extent to multiple classifications using diversity without multiple classifiers can improve classification accuracy.
- The design of a Multiple Classification Scheme that is not constrained by either a classifier boundary or the number of classifiers.

The Algorithm described in this section is a new approach to classification problem with an ensemble of classifiers. Instead of use multiple classifiers, it uses a single one and creates in the dataset some new samples 'generated' from original dataset. Using a single classifier, it achieves multiple classification and so benefits from the logic of a multiple classification scheme without repetitive classifier training and without classifier selection.

proposed algorithm

The method proposed in [6] is based on the idea of the generation of some artificial samples from training set. These samples are called *pseudo points* and introduce diversity in data. Then the new dataset is used to train a classifier. Given a test sample and some training samples, proposed method divides the training samples into two groups: one containing reference samples, and the other containing evaluation samples. Different reference samples are used to generate different pseudo test data points, each of which constitutes a different combination of an original test sample and some reference samples (Figure 3.5). In a second phase evaluation samples are used to select adequate reference samples for pseudo test data point generation.

Using the approach described above, in the following named SMCS (Single classifier-based Multiple Classification Scheme), diversity in dataset is introduced from the original training set using different reference samples to create *pseudo points*. In Figure 3.5 is represented a schematic view of the process of SMCS.

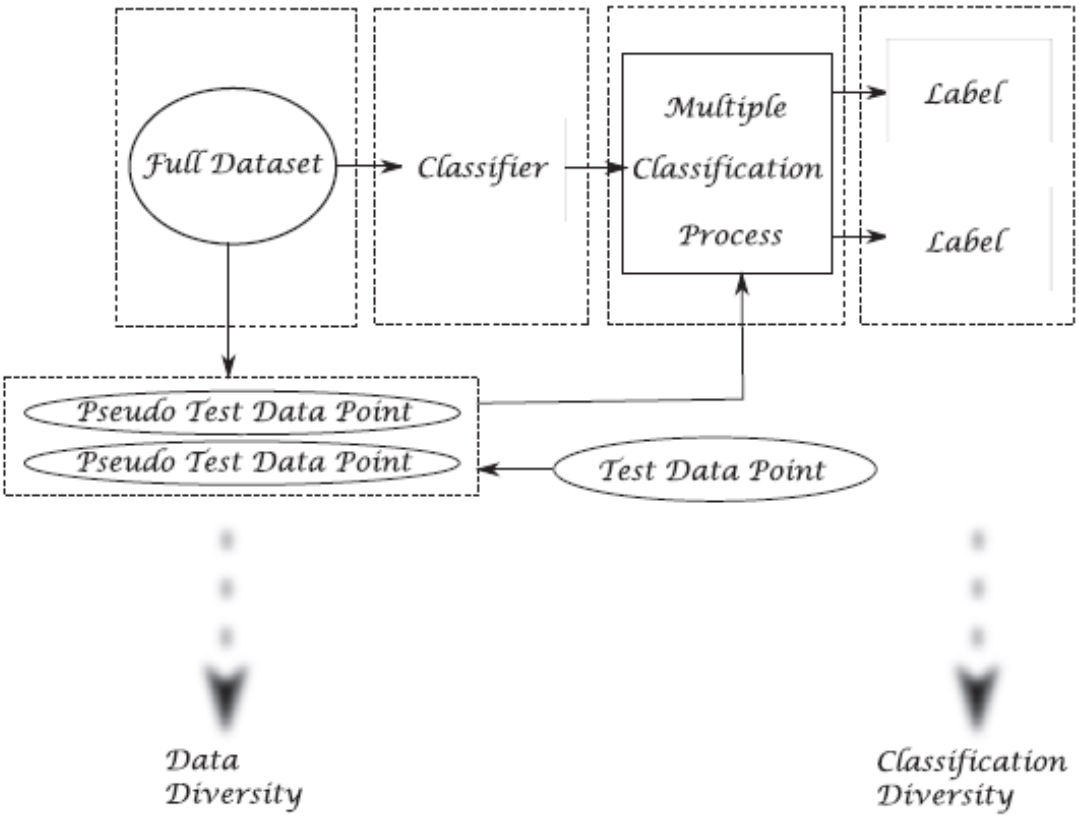


Figure 3.5 Schematic representation of phases of SMCS algorithm.

In the next paragraph, will be illustrated some advantages of SMCS respect to Multiple Classification Scheme (MCS), which represent an approach based on an ensemble of classifiers.

Initially, considering how the diversity in classification is generated, it can be notice that MCS benefits from the fact that each classifier has a different perception of how a test sample should be classified. Because the decision boundary calculated by each classifier is different, there is diversity among decision boundaries drawn by different classifiers. Given that classifiers make different errors on different test samples, diversity can actually help to improve classification accuracy. So, in the MCS, one of the core issues is to generate, select and combine multiple classifiers, such that the combined decision boundary is better than any existing single boundary.

The SMCS algorithm tries to find a decision boundary with the potential to be close to the oracle and not constrained by an existing classifier boundary or the number of classifiers. The design of SMCS, must consider the fact that a decision boundary drawn by a classifier might never be optimal. For this reason, instead of refining several existing decision boundaries by combining them together, SMCS tries to explore and make use of information in the neighborhood of a single decision boundary. In this way, this approach looks for diversity that is already present in the neighborhood, rather than trying to benefit from diversity embedded in different classifiers. Consequently, diversity is extracted not from diverse decision boundaries, but from diverse pseudo test data points.

The core issue is then to adequately generate, select, and combine multiple pseudo test data points for a test sample, rather than generating, selecting, and combining multiple classifiers. The proposed SMCS approach focalize on three principal problems:

- How extract diversity from a dataset directly without training multiple classifiers?
- Can multiple classification without multiple classifiers enhance classification accuracy?
- How can SMCS performance be compared with MCS performance?

algorithm description

Firstly the training set $\boldsymbol{X}$ is divided in N reference samples $\boldsymbol{X_r} = \{x_1, x_2, \dots, x_N\}$ and M evaluation samples $\boldsymbol{X_e} = \{x_1, x_2, \dots, x_M\}$. A single classifier C_x is trained and is considered a single test point $\tilde{x_t}$. The mechanism involves the creation of K pseudo test data points $\widehat{\boldsymbol{X}}_{\boldsymbol{t}} = \hat{x}_{1,t}, \hat{x}_{2,t}, \dots, \hat{x}_{K,t}$, which would result in K corresponding classification outputs $\hat{y}_{1,t}, \hat{y}_{2,t}, \dots, \hat{y}_{K,t}$ after being classified by C_x. The purpose of this mechanism is to generate $\widehat{\boldsymbol{X}}_{\boldsymbol{t}}$, such that the combination of classification outputs on these K pseudo test data points $\hat{y}_t$ will be as close to the true class label y_t as possible. Note that:

$$\hat{y}_t = g(\hat{y}_{1,t}, \hat{y}_{2,t}, \dots, \hat{y}_{K,t}) \tag{3.14}$$

where $g(\cdot)$ is a classification combination function, such as majority voting.

There exist many way to calculate the pseudo points: a pseudo test data point can be generated as a combination of a test sample and a reference sample, or as a combination of a test sample and several reference samples. It can be generated in a deterministic way, or with some random factor. In this implementation, as in the reference article [6], pseudo data points are generated from test sample and reference samples considering the features value and calculating an average between features value of test sample x_t and reference sample x_r.

For example, if each data point has L feature, the feature l of the generated pseudo test data point $\hat{x}_{i,t}$ will simply be a weighted average of the same feature of the test sample $\tilde{x}_{i,t}$ and the corresponding feature of reference sample x_i:

$$\hat{x}_{i,t,l} = \alpha x_{i,l} + (1-\alpha)\tilde{x}_{t,l} \qquad 1 \leq l \leq L\,, 0 \leq \alpha \leq 1 \qquad (3.15)$$

where $\hat{x}_{i,t,l}$ indicates the value of the feature l of the generated pseudo test data point $\hat{x}_{i,t}$. $x_{i,l}$ indicates the value of the feature l of the reference sample x_i and $\tilde{x}_{t,l}$ indicates the value of the feature l of the test sample $\tilde{x}_{i,t}$. Also note that α controls the noise and diversity in pseudo test data points: the larger it is, the greater the diversity and the noise.

However not every pseudo data point generated by test sample and reference samples will be considered a valid point to add to the dataset. To determine which point will be adequate for classification, evaluation samples will be used to determine good reference sample to use. In particular the algorithm implements the following steps:

1) For every reference sample, use the M evaluation samples to determine if it's good for classification. With the procedure used to determine pseudo point, find the pseudo points relative to the reference sample and each evaluation sample and use classifier C_x to classify the pseudo points. If the classification of this pseudo data point has the same label as the evaluation sample, then this [evaluation sample–reference sample] pair is regarded as valid; otherwise, it is regarded as invalid.
2) Assign weight to reference samples. For a given test sample, we find the m nearest evaluation samples. Then, to every reference sample is assigned a weight based on its validity with respect to these m evaluation samples, which is obtained as the sum of the relatives [evaluation sample–reference sample] valid pairs. Weight represents the "goodness" of the reference sample in the dataset.
3) Select reference samples and generate pseudo test data points. A threshold for the reference samples is defined, and only reference samples with weights higher than that threshold are selected for pseudo test data point generation.

identify valid [evaluation sample–reference sample] pairs

Between each reference and evaluation sample, a pseudo data point is generated considering equation (3.14) and then a validity measure $v_{i,k}$ is defined for each [evaluation sample $\tilde{x}_k$– training data point x_i] pair, $1 \leq k \leq M, 1 \leq i \leq N$ and the validity is set to 1 for correct classification and to 0 for incorrect classification:

$$v_{i,k} = 1, \quad if\ \hat{y}_{i,k} = y_k \tag{3.16}$$
$$v_{i,k} = 0, \quad otherwise$$

assign weight to reference samples

Given a test sample $\tilde{x}_t$, consider the nearest m evaluation samples to be trustworthy for this test sample, noting that $m \ll M$. m evaluation data points are then used to evaluate the fitness of reference samples for the test sample $\tilde{x}_t$. The weight of a reference sample x_i is assigned as follow:

$$w_i = \sum_{k=1}^{m} \delta_{i,k} v_{i,k} \tag{3.17}$$

where $v_{i,k}$ is a validity measure of the [evaluation sample $\tilde{x}_k$– reference sample x_i] pair, and $\delta_{i,k}$ is a weighting adjustment based on distance or other factors. In this work, as in the reference paper [6], the weighting adjustment $\delta_{i,k}$ is defined as:

$$\delta_{i,k} = \frac{d(\tilde{x}_k, \tilde{x}_t) + d(\tilde{x}_k, x_i)}{d(\tilde{x}_t, x_i)} \tag{3.18}$$

where $d(\cdot)$ indicates an Euclidean distance function, $\tilde{x}_k$ is an evaluation sample, x_i is a reference sample, and $\tilde{x}_t$ is a test sample.

select reference samples and generate pseudo test data points

Given a test sample $\tilde{x}_t$, only the nearest n reference samples are evaluate for the test sample. Is also defined a threshold θ. Therefore, the selection criterion for reference samples is:

$$\begin{aligned} &if\ w_i \geq \theta && s_i = 1 \\ &else && s_i = 0 \end{aligned} \tag{3.19}$$

where s_i is the selection decision on reference sample x_i. The threshold θ is defined as:

$$\theta = \rho \max\{w_i\} \qquad 0 < \rho \leq 1 \tag{3.20}$$

combine multiple classification outputs

Consider pseudo data points relatives to test point $\tilde{x}_t$: $\hat{x}_{i,t}$, $1 \leq i \leq I$ where I is the number of generated pseudo points. At each pseudo data point is assigned a label $\hat{y}_{i,t}$. Then the label assigned to test point $\tilde{x}_t$ is determined using a fusion function g of labels of pseudo points:

$$\hat{y}_t = g\left(\hat{y}_{1,t}, \hat{y}_{2,t}, \dots, \hat{y}_{I,t}\right) \tag{3.21}$$

In the code implementation of this thesis work, function g is calculated considering labels of pseudo points, weighted with their *decision value (*a measure of distance from the boundary provided by libSVM).

SMCS - Pseudoocode

Input: labelled data, test sample
Output: label of test sample: $\hat{y}_t$
Method:

Define reference dataset $X_r = \{x_1, x_2, \dots, x_N\}$, evaluation dataset $\tilde{X}_e = \{\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_M\}$ and a test sample $\tilde{x}_t$
Define pseudo data point generation function $f: \tilde{x}_t \mapsto \hat{x}$

Train a single classifier C_X using all reference samples and evaluation samples

Identify valid [evaluation sample–reference sample] pairs:
for all evaluation samples $\tilde{x}_k : k = 1, \dots, M$ **do**
for all reference samples $\tilde{x}_i : i = 1, \dots, N$ **do**
Generate pseudo data point $\hat{x}_{i,k}$ with function f and reference sample x_i for evaluation sample $\tilde{x}_k$
Classify pseudo data point $\hat{x}_{i,k}$ with classifier C_X:
if Classification output $\hat{y}_{i,k}$ of pseudo data point $\hat{x}_{i,k}$ equals the class label $\tilde{y}_k$ of evaluation sample $\tilde{x}_k$ **then**
Set validity $v_{i,k}$ of [evaluation sample $\tilde{x}_k$ –reference sample x_i] pair to be 1
else
Set validity $v_{i,k}$ of [evaluation sample $\tilde{x}_k$ –reference sample x_i] pair to be 0
end if
end for
end for

Assign weight to reference samples for test sample $\tilde{x}_t$:
for all reference samples $x_i : i = 1, \dots, N$ **do**
Initialize ω_i of reference sample x_i to be 0
for Nearest m evaluation samples from test sample $\tilde{x}_t, x_k: k = 1, \dots, N$ **do**
Evaluate the weight of reference samples x_i for test sample $\tilde{x}_t$:
if validity $v_{i,k}$ of [evaluation sample $\tilde{x}_k$ –reference sample x_i] pair equals to 1 **then**
Increase the weight ω_i of reference samples x_i: $\omega_i = \omega_i + \delta_{i,k}\, v_{i,k}$
end if
end for
end for

Select reference samples and generate pseudo test data points:
for Nearest n reference samples from test sample $\tilde{x}_t, x_i : i = 1, \dots, n$ **do**
if the weight ω_i of reference samples x_i is greater than the threshold θ **then**
Generate a pseudo test data point $\hat{x}_{i,t}$ for test sample $\tilde{x}_t$ based on reference sample x_i:

$$\hat{x}_{i,t} = f(\tilde{x}_t, x_i)$$

Classify pseudo data point $\hat{x}_i$ with classifier C_X, and store the classification output $\hat{y}_i$
end if
end for

Apply a fusion function g to combine multiple classification
outputs: $\hat{y}_t = g(\{\hat{y}_{i,t}\})$
Return combined classification output $\hat{y}_t$ as final result.

Algorithm 3.5 semi-supervised features ranking.

Chapter 4

Experimental results

In this chapter it will be analyzed the structure of tests for the developed algorithms based on the reference articles discussed in Chapter 3. The chosen datasets for this tests will be described in section 4.2 and will be compared to datasets used for the experiments in the various articles, especially the paper [7] "*DCPE co-training for classification"* which will be also used as reference framework for experimental dataset and division into training set, test set and validation. In the chapter will be also presented the validation protocol (a 5 fold cross validation) and its implementation, the approach to multiclass problem (using one vs all approach to reduce the multiclass problem to binary class case) and the comparison schema for comparing the developed algorithm performance to a baseline supervised method (a Support Vector Machine).

An important result will be the high performance of semi-supervised learning (also in its simplest base form of self-training) respect to supervised learning. The limited number of labeled data, makes desirable the use of unlabeled data to generate a classifier, and this can greatly improve the performance, as will be seen in experimental results of test 3.

The programs are implemented in MATLAB® 2013a on Windows 7® home premium 64 bit.

4.1 Used configuration framework for data

The adopted configuration for tests is the framework described in [7] in paragraph 4.1.

As a framework for test experiments is used a data division as shown in Figure 4.1. Original dataset is divided in two distinct subsets respectively for training (80%) and for testing (20%). A protocol of 5-fold cross validation is adopted. In this protocol the process of training and test is repeated 5 time on 5 different folds.

Each fold is divided in test set and training set; training set is again splitted randomly into three groups: validation set (10%), labeled data (10%) and unlabeled data (80%). The resulting data structure, which will be used for experiment, has the form illustrated in Figure 4.2. In each iteration of cross validation process, is considered a row of structure represented in Figure 4.2, which is a fold of distinct elements of the dataset.

Each sampling phase is realized without resampling, obtaining as results sets of different data distribution, which can simulate different datasets of the same kind.

Finally it's important to note that sampling is implemented maintaining the original class distribution of the entire dataset. The sampling process that mantain the class distribution can be implemented respectively whit resampling (not used in this phase of dataset splitting) and without resampling.

Figure 4.1 Data splitting schema used for experiments.

Fold number [1-5]	test set 20% of datasets	validation set 10% of training set	labelled data 10% of training set	unlabeled data 80% of training set

Figure 4.2 the datasets splitting for cross validation. Each line contain the data of relative fold.

4.1.1 Validation phase

The validation data is used to test the classifier obtained with the tested algorithms, in each step of cross validation.

The variable to optimize depends on the used classification algorithm. In the executed tests, the algorithms which are subject to cross validations are the features ranking algorithm, the DCPE algorithm and the clustering-based algorithm. In these cases the variable to optimize in validation phase are respectively the following:

- The number of features to consider.
- The number of iterations.
- Threshold ε_1, as described in paragraph 3.1.2.

In DCPE algorithm, the threshold ε_1 represent the confidence level of the semi-supervised classification with clustering-based fuzzy c-mean algorithm. This threshold is progressively incremented in validation phase and the value which provides better performance is selected for tests.

The validation methods implemented are described in the reference articles presenting the algorithms discussed above:

[8] *A semi-supervised feature ranking method with ensemble learning*

[7] *DCPE co-training for classification*

[9] *Using clustering analysis to improve semi-supervised classification*.

4.2 Datasets

Datasets considered for experiments are the datasets available on the internet websites of UCI (University of California, Irvine) in the archive dedicated to machine learning: http://archive.ics.uci.edu/ml/, commonly used in many research experiments.

Tested datasets are quite all datasets used in experiments in [7] (section 4). For some technical problems, y*east* dataset is not used because it has a problematic class imbalance, which not allows to divide datasets maintaining class distribution. For the tests proposed in this thesis work, this problem has not be managed and for simplicity dataset y*east* is not used.

Similarly, dataset Letter, consisting of features describing black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet, has not been used for consumption time problems. Due to the high number of classes (26, one for each possible letter), the one-vs-all conversion from multiclass to binary problem causes an high computing time consumption and so, for practical reasons, this datasets is not tested in the experimental phase.

The used datasets are listed in Table 4.1, specifying for each dataset the number of samples, the features dimension and the class distribution. These dataset's characteristics will be take into account when considering class imbalance problem. The datasets source also specify which attribute is continuous, but we do not consider this parameter in data elaboration.

In Table 4.2 is reported a brief description of datasets and the link to UCI website where used data are available.

Dataset name	Total size	Number of attributes	Number of classes	Class distribution
Tictactoe	958	9	2	∽65.3% positive
Vowel	990	10	10	Equally distributed
Car	1728	6	4	Unacc 1210 (70.023 %) acc 384 (22.222 %) good 69 (3.993 %) v-good 65 (3.762 %)
Credit	690	15	2	+ 307 (44.5%) - 383 (55.5%)
Ionosphere	351	34	2	g:126 b:225
Pendigits	3498	16	10	0: 363 1: 364 2: 364 3: 336 4: 364 5: 335 6: 336 7: 364 8: 336 9: 336
Pima	768	8	2	0 500 1 268
Segmentation	2100	19	7	300 instances per class.
Sonar	208	60	2	M (metal cylinder) 111 R (rock) 97
Australian	690	14	2	+ 307 (44.5%) - 383 (55.5%)
WPBC	198	34	2	Nonrecur 151 Recur 47

Table 4.1 Datasets used in experiments.

Dataset name	Brief description	link
Tictactoe	Binary classification task on possible configurations of tic-tac-toe game	https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame
Vowel	Speaker independent recognition of the eleven steady state vowels of British English using a specified training set of lpc derived log area ratios. Note: used data are in the form of reformulated vowel-context, available at the link, provided by Peter Turney, peter@ai.iit.nrc.ca	https://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/connectionist-bench/vowel/vowel-context.data
Car	Derived from simple hierarchical decision model, this database may be useful for testing constructive induction and structure discovery methods.	https://archive.ics.uci.edu/ml/machine-learning-databases/car/car.data
Credit	credit card applications; good mix of attributes	https://archive.ics.uci.edu/ml/machine-learning-databases/credit-screening/crx.data
Ionosphere	Classification of radar returns from the ionosphere	https://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/ionosphere.data
Pendigits	Digit database from 44 writers	https://archive.ics.uci.edu/ml/machine-learning-databases/pendigits/pendigits.tes
Pima	From National Institute of Diabetes and Digestive and Kidney Diseases.	https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data
Segmentation	Image data described by high-level numeric-valued attributes, 7 classes	https://archive.ics.uci.edu/ml/machine-learning-databases/image/segmentation.test
Sonar	The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock.	https://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/connectionist-bench/sonar/sonar.all-data
Australian	Credit card applications. This database exists elsewhere in the repository (Credit Screening Database) in a slightly different form.	https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/australian/australian.dat
WPBC	Prognostic Wisconsin Breast Cancer Database	https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data

Table 4.2 Datasets used in experiments: description and link.

4.3 Prepruning and considerations

4.3.1 Initial Datasets Elaboration

Initially datasets have been downloaded from the UCI website in txt extension.

Data were converted from original text document to a valid format for MATLAB input. Text lines have been delated or commented (inserting "%" character at the beginning of the line).

Then, using a simple script in Java, categorical attributes represented by letters has been converted in a numerical representation. This operation introduced an order in the feature space that was not always implied from the previous representation with letters. This was not considered a problem since categorical values of attributes are considered separately.

Some datasets presents missing values. The samples containing missing values was deleted from datasets since, for the purpose of this thesis work, performance in presence of missing values or a robust model of classification are not take into account. Tests of the algorithms considering missing values could be a future development of the approach to machine learning presented in this thesis.

Finally in some datasets such as *WPBC* has been necessary to delete attributes that aren't useful for classification, since they don't add information regarding data.

For example the ID attribute has been deleted because other attributes and label are totally independent from it. Another example is the first attribute in vowel dataset in contextual form. This attribute concerns about the usage for training or test in previous experiments and is uncorrelated with label and data.

4.3.2 Data Normalization

Datasets attributes has to be normalized for an efficient classification using libSVM. For the normalization each attribute is reported in the [-1,1] interval. The classes are also reduced to the integer $0, \dots, c$ where c is the number of classes. The pseudocode used to normalize data is the following:

Normalization pseudocode
Input: A: arbitrary array
Output: NA: normalized array
Method:
1. M= maximum value in A
2. m= minimum value in A
3. **for each** element A(i) in A
4. NA(i)=(A(i)-m)/(M-m)
5. **end**

Algorithm 4.1: pseudocode for normalization.

The normalized values are determined in accord with the following formula:

$$A_i{}^N = \frac{A_i - \min A_j}{\max A_j - \min A_j} \qquad (4.1)$$

Where $A_i{}^N$ is the normalized attribute value of record i, A_i is the attribute A of record i, and $\max A_j$ and $\min A_j$ represent respectively the maximum and minimum value of the attribute A among all records $j = 1, \dots, N$ where N is the number of considered records.

It’s important to notice that training set and test set are normalized separately. This is important for the coherence in the training set and test set definition: indeed, training set can’t be used to modify test set in any way and vice versa.

4.3.2 Considerations and details about datasets

In this paragraph, some characteristics of tested datasets will be shown in more detail. In particular, referring to the class imbalance problem and overfitting:

- **Tictactoe:** This database encodes the complete set of possible board configurations at the end of tic-tac-toe games, where "x" is assumed to have played first. The target concept is "win for x" (i.e., true when "x" has one of 8 possible ways to create a "three-in-a-row"). Interestingly, this raw database gives a stripped-down decision tree algorithm (e.g., ID3) fits. However, the rule-based CN2 algorithm, the simple IB1 instance-based learning algorithm, and the CITRE feature-constructing decision tree algorithm perform well on it [31].
- **Vowel:** Speaker independent recognition of the eleven steady state vowels of British English using a specified training set of derived log area ratios. In the used form, data has been contextualized: implicit in the original data is contextual information on the speaker's gender and identity. In the contextualized file (used for this thesis work), the speaker's gender and identity have been added as new features [31].
- **Car:** Car Evaluation Database was derived from a simple hierarchical decision model. Because of known underlying concept structure, this database may be particularly useful for testing constructive induction and structure discovery methods. It presents also class imbalance characteristic [31].

- **Credit:** This file concerns credit card applications. All attributes, names and values have been changed to meaningless symbols to protect confidentiality of the data. This dataset is interesting because there is a good mix of attributes: continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values [31].
- **Ionosphere:** The dataset consists in parameter measured by radar that investigated the ionosphere.

  Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this dataset are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal.

  The high number of features can cause an overfitting problem in some learning algorithm [31].
- **Pendigits:** The dataset consists in a collection of 250 samples from 44 writers. The samples written by 30 writers are used for training, cross-validation and writer dependent testing, and the digits written by the other 14 writers are used for writer independent testing. This dataset is also available in the UNIPEN format. The original dataset available online is composed of two parts: one for training and one for test, containing digits from two separated set of writers. In this thesis work, the considered dataset is only the test one [31].
- **Pima:** It contains data about diabetes outbreak in Pima Indian population. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. This dataset presents also a class imbalance problem, since the number of samples of positive class is about half of the ones with negative class [31].
- **Segmentation:** The instances were drawn randomly from a database of 7 outdoor images. The images were segmented to create a classification for every pixel. Each instance is a 3x3 region [31].

- **Sonar:** Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occurs later in time, since these frequencies are transmitted later during the chirp. The number of features is very high and so the risk of overfitting is relevant [31].
- **Australian:** This dataset is similar to Credit dataset and concerns credit cards approval for Australian citizens [31].
- **WPBC:** Each record represents follow-up data for one breast cancer case. These are consecutive patients seen by Dr. Wolberg since 1984, and include only those cases exhibiting invasive breast cancer and no evidence of distant metastases at the time of diagnosis. This dataset presents a class imbalance: the "recur" class is about one third of "nonrecur" class [31].

4.4 Tests schemas

In this section, the tests that have been effected, the structures of algorithms and how they are combined together will be presented. In particular in test 5 the DCPE and clustering-based algorithm are used, instead that with SVM as base classifier (test 4), with an implementation of *Single Classifier-based Multiple Classification Scheme* presented in [6]. Test 1 and test 2 consist in dividing the datasets, normalizing them and applying a baseline SVM classifier. The accuracy of SVM will be used as baseline for later comparison. Test 3 is an application of feature selection in a semi-supervised implementation [11], which allows also to see the Self-training performance. Test 4 consists in the test of DCPE and clustering-based learning algorithms with the base classifiers (SVM, ADABoots and Semi-Supervised Fuzzy c-Means).

4.4.1 Test 1

Initial test consists in the initial step of import datasets in MATLAB, normalization of attributes and standardization of classes.

For each dataset, the samples are divided in training set and test set following the protocol described in section 4.1. It's important to specify that data has been elaborated using a normalization function, as described in paragraph 4.3.2, and a selection function which respects class distribution, as said in paragraph 4.1.

It's important to note that, for practical aspects, the sampling maintaining the class distribution can't be used in that datasets which have not enough elements of any class. Since the smallest subset considered (validation set or labeled data set) is composed of 8% of the total samples, the number of elements in each class has to be more then 13 to guarantee that each subset has at least an element of each class. This is the reason why *yeast* dataset is not considered in experiments.

4.4.2 Test 2

Test 2 consists in the baseline test. In this test, the classification of test set data is experimented using labeled.

Unlabeled and validation data are not used in the baseline classification, because the mean of this analysis is to compare advanced techniques proposed in Chapter 3 to a standard supervised learning classification. Since baseline is supervised, it can't use unlabeled data and has to reduce its learning set to labeled data (which has a limited size, considering the dataset division).

For the SVM in this baseline test, the following libSVM parameters are used:

- - g 1 : γ in the kernel function is set to 1.
- - c 100 : a cost parameter is used, also to limit inaccuracy due to class imbalance in multiclass problem with one-vs-all approach.
- -t 2 : kernel function is of radial basis type.

In this section it's important to notice the implementation of one-vs-all approach, which is used to convert a multiclass classification problem into a binary class one. This approach has already been explained in 2.2.5. For the practical implementation, is used a *for loop*, in which iteratively labels are set to 0 if the classes are different from i, while if classes are equal to i is set to 1. Then in each loop the classifier is built using the training set, and the test samples are classified. To each test sample is also assigned a decision value. In this case of a simple SVM, the decision value is equal to the distance from the sample to the boundary. This value is provided by libSVM library. The decision values are progressively inserted in a matrix $M = k * c$, where k is the number of test samples and c the number of classes. The final class assigned to the test sample is the class associated to the column of M relative to x with the highest value.

The Algorithm 4.1 shows the pseudocode for test 2 program, while Algorithm 4.3 describes the implementation of one-vs-all approach. Results of the baseline classification with cross validation are presented in Table 4.3.

BASELINE classification script

Datasets= $(dataset_1, \dots dataset_n)$
Insert in the cell arrays Data and Labels the samples in each dataset
For each dataset
    Apply DivideDataset and obtain the structure of Figure 4.2
    Results= TestSVM(Labelled set, Test set)
End
Save Results

Algorithm 4.2 script used for the baseline tests.

BASELINE one-vs-all approach – TestSVM.m

Input: training set, test set
Output: Accuracy, decision values, confusion matrix

Decision_values_final=∅
Accuracy=∅
confusion matrix=∅
Predict_labels=∅
For each class $i = 1, \dots c$
    $label(i) = i$ are substituted with 1 , $i = 1, \dots, n$
    $label(i) \neq i$ are substituted with 0 , $i = 1, \dots, n$

    Train SVM ϕ on test set
    (predict_label, decision_value_temp) =svmpredict (test set, ϕ)
    Decision_values (:,i)= decision_value_temp
End
Predict_labels(i) is set to the max index of column of Decision_values(i,:)
Decision_values_final(i) is set to the maximum value of Decision_values(i,:)
For each test sample x_i
    If (Predict_labels(i) = $label(i)$)
        Increment Accuracy
End
For each class couple $(i, j),\ i, j = 1, \dots, c\ ,\ i \neq j$
    for each test sample x_k
        if (label(k)=i and predict_labels(k)=j)
            increment confusion_matrix(i,j)
    end
end

Algorithm 4.3 baseline classification with one-vs-all approach.

Dataset name	Baseline SVM accuracy
Tictactoe	0.7179
Vowel	0.5111
Car	0.8921
Credit	0.8124
Ionosphere	0.6522
Pendigits	0.9775
Pima	0.6868
Segmentation	0.9257
Sonar	0.6800
Australian	0.8175
WPBC	0.9214

Table 4.3 Results of baseline SVM.

4.4.3 Test 3

In this test phase, the purpose is to apply feature selection and see how accuracy can be increased considering a smaller number of features.

Moreover, after the selection of the optimal features, a semi-supervised learning algorithm is applied (in this implementation, the Self-training algorithm is used, as described in reference article [8]). The classification with semi-supervised learning is also described in the original algorithm proposed in paper [8] *"A semi-supervised feature ranking method with ensemble learning"*. This classification

technique, as widely explained in section 2, can generally increase the performance of the classifier considering also unlabeled data, which are more than labeled data (following the protocol of data division, presented in [7] and explained in section 4.1, the unlabeled data are 80% of training set, while labeled data are only 10% of training data).

Following the schema described in test 2 (paragraph 4.4.2), the following steps are applied:

1. Cross validation loop.
2. One-vs-all implementation.
3. Features ranking algorithm (for binary problem).

The schema in Figure 4.3 represents a diagram of function calls.

Feature selection test phase consists of two phases: initially the best features are selected following the procedure described in section 3.1.3. In the second phase the validation data are classified using Self-training and both labeled and unlabeled data, but considering only the optimal number of features. This test provides in output the optimal subset of features.

Then the test set is classified with a Self-training approach, which constructs a classifier using labeled and unlabeled data. This classification of test set is executed considering only the optimal features subset.

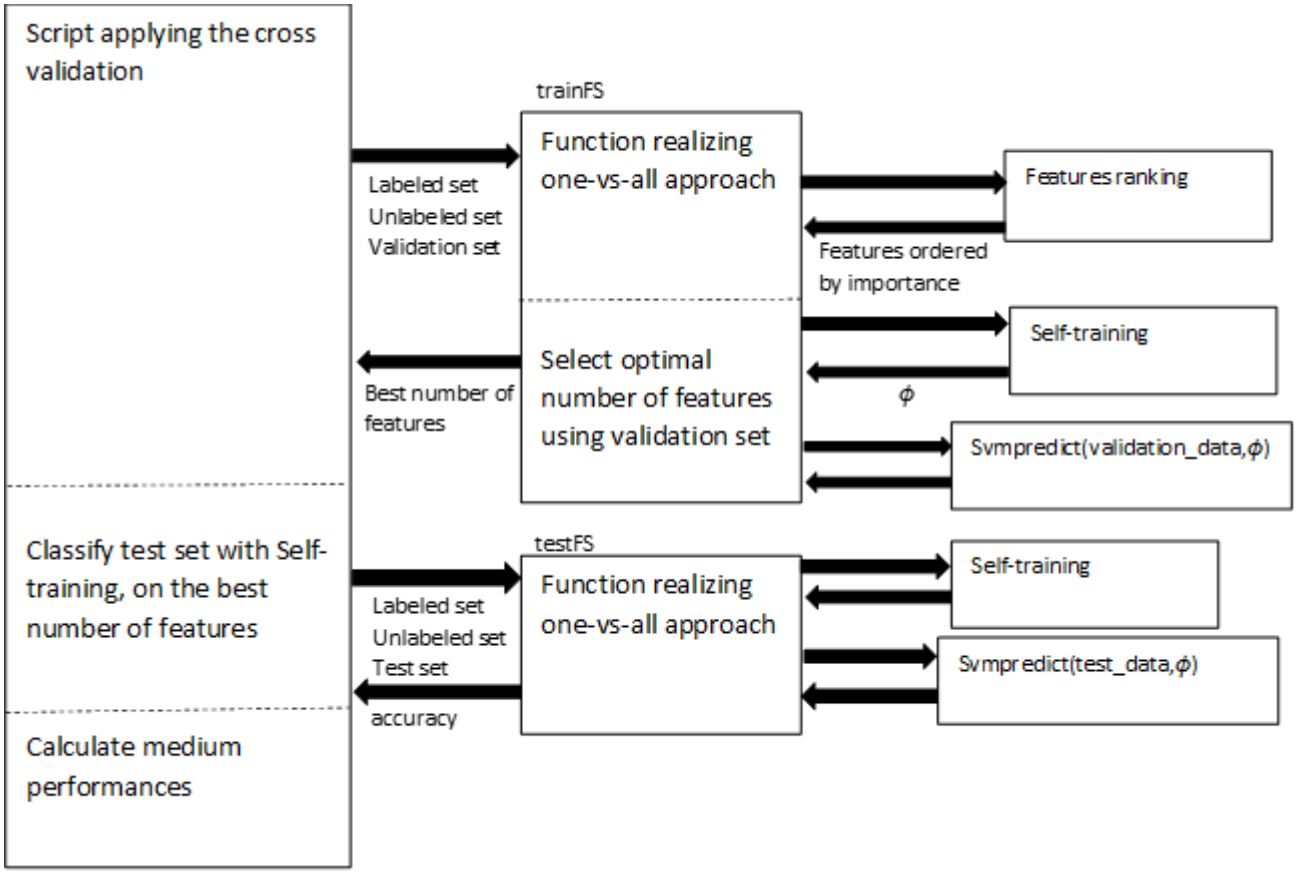
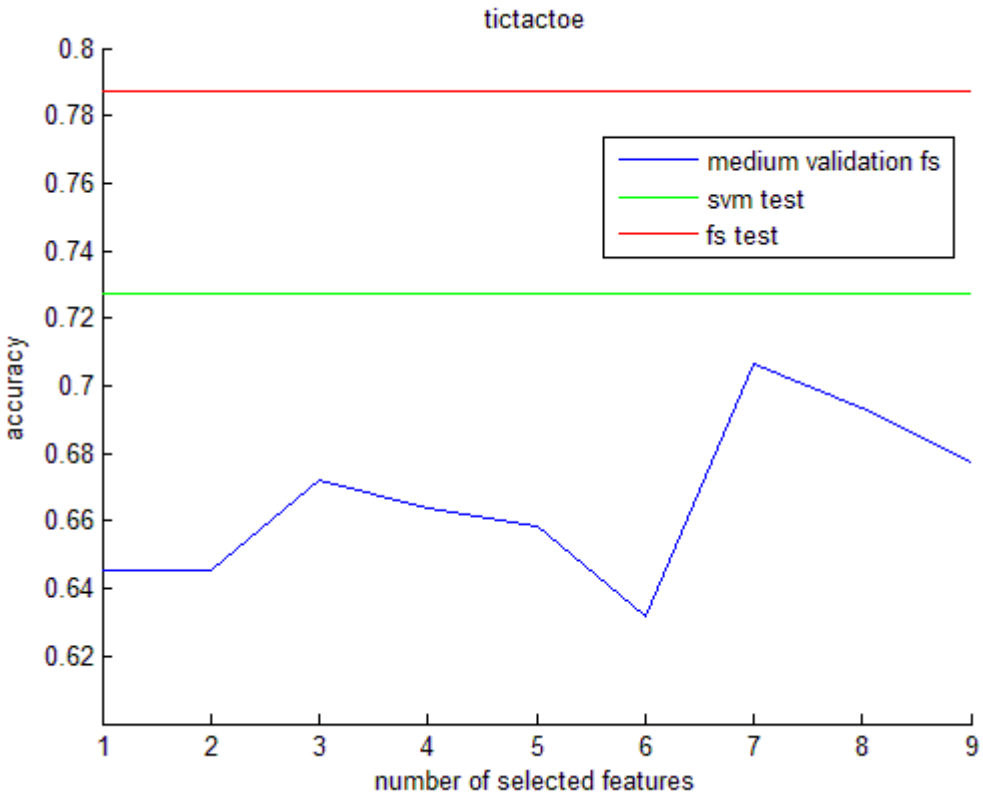
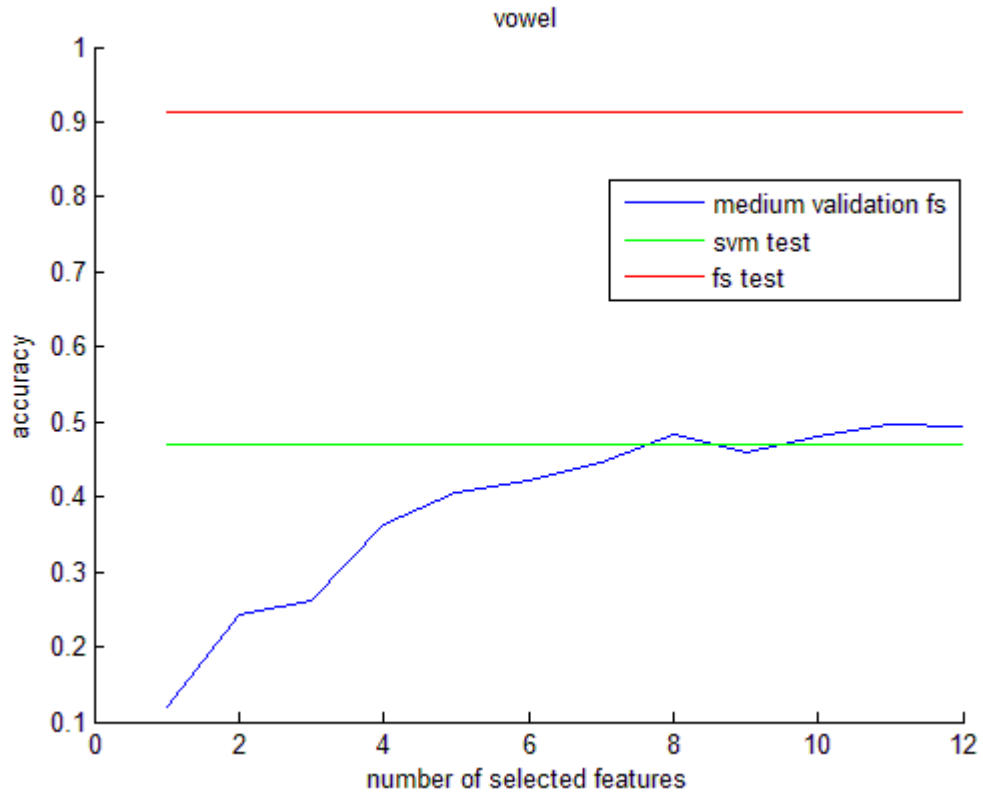


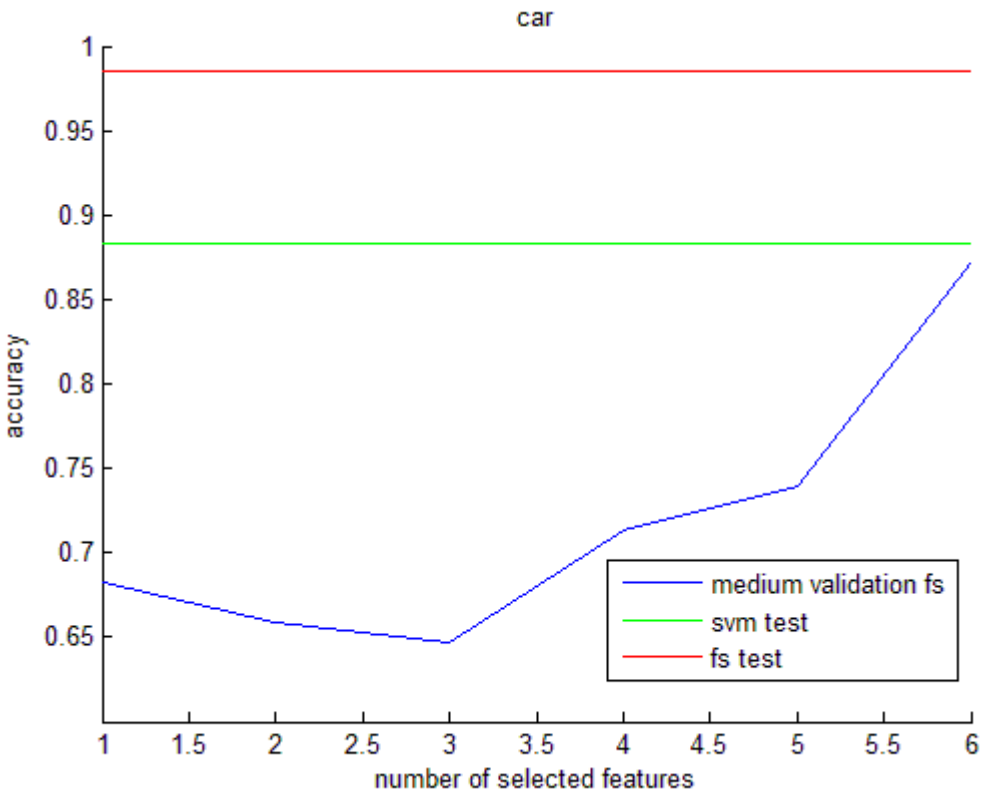
Figure 4.3 schema of test 3.

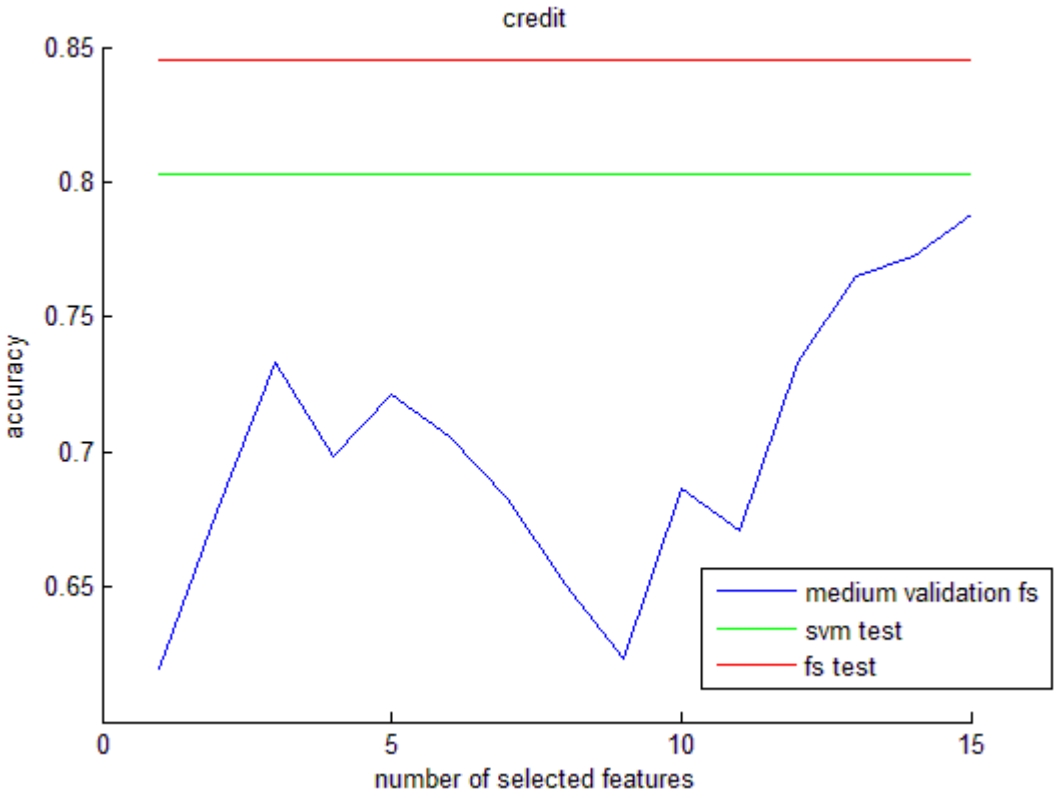
results of test 3

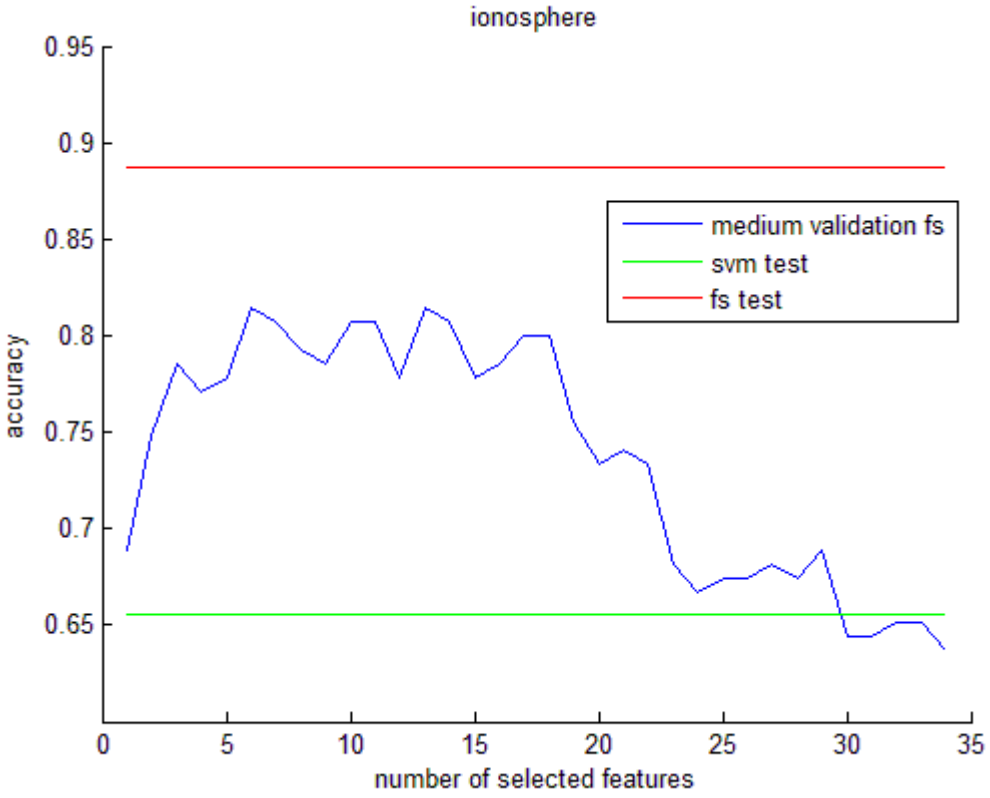
Results show how accuracy varies increasing the number of selected features. Graph are drawn calculating the mean of the cross validation steps. Then the results in terms of final accuracy on test set are provided in Table 4.4.

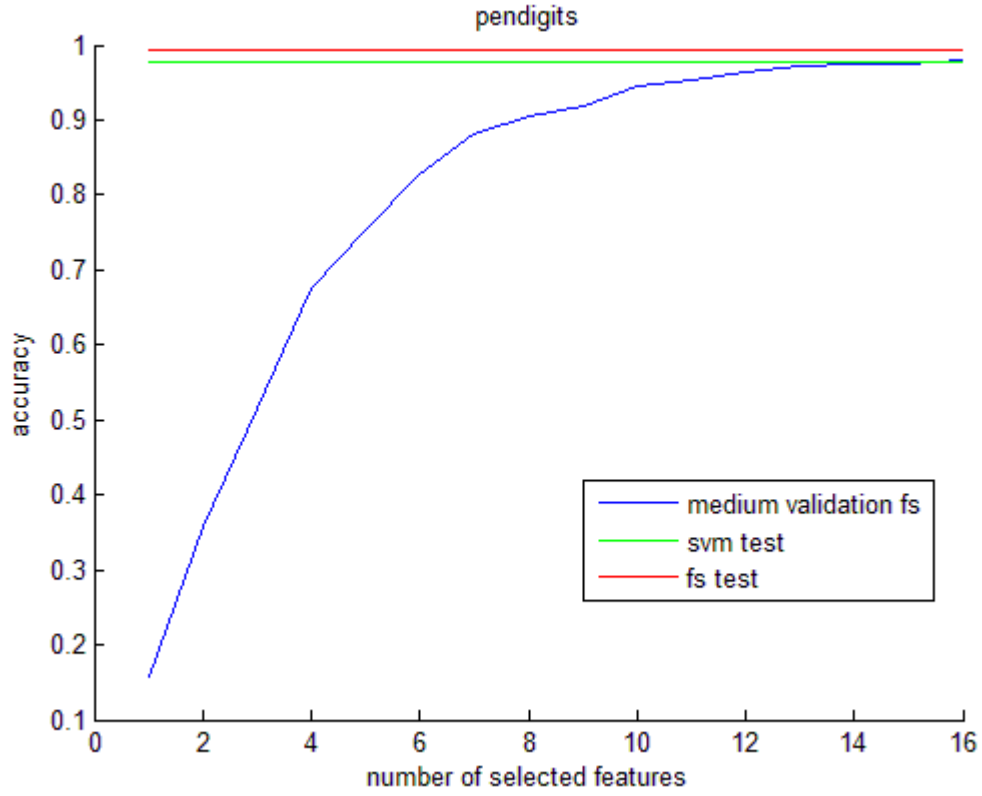

tictactoe
0.8
0.78
0.76
0.74
0.72
0.7
0.68
0.66
0.64
0.62
accuracy
1
2
3
4
5
6
7
8
9
number of selected features
medium validation fs
svm test
fs test

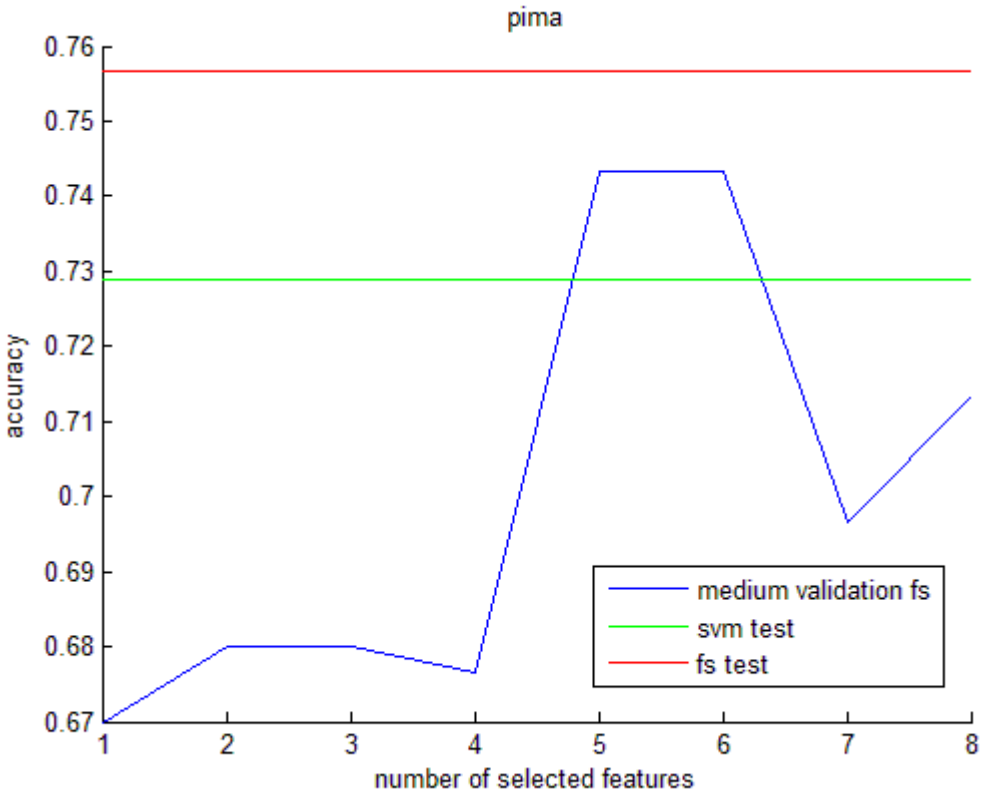

vowel
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
accuracy
0
2
4
6
8
10
12
number of selected features
medium validation fs
svm test
fs test

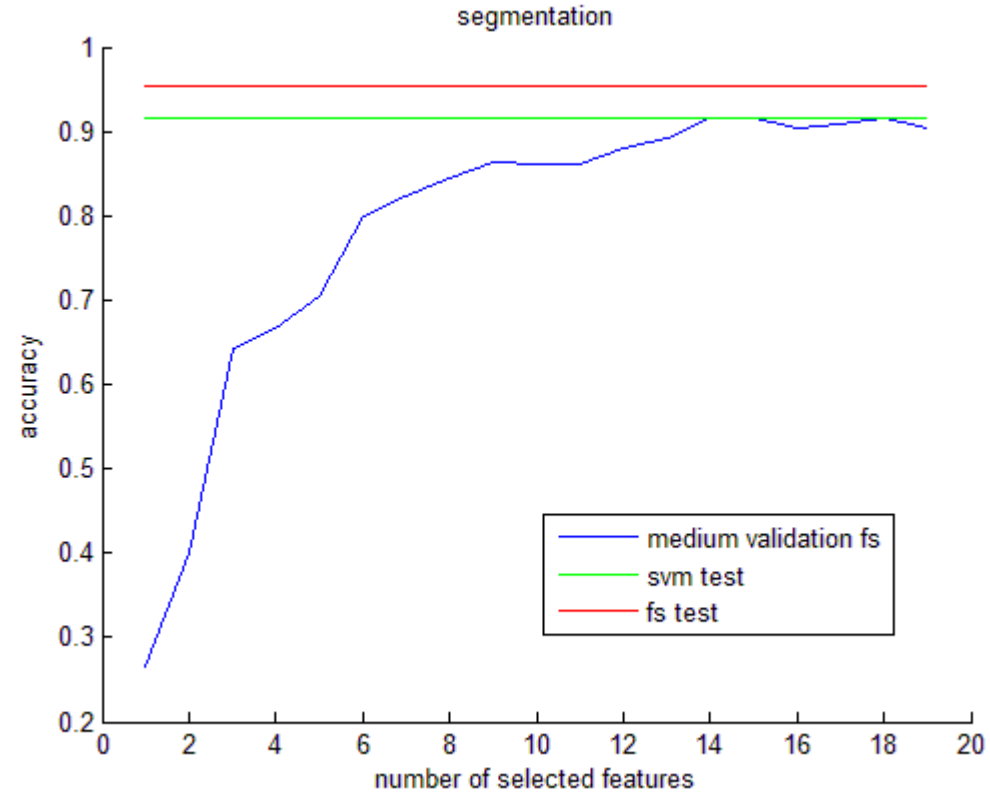

car
accuracy
number of selected features
medium validation fs
svm test
fs test
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6

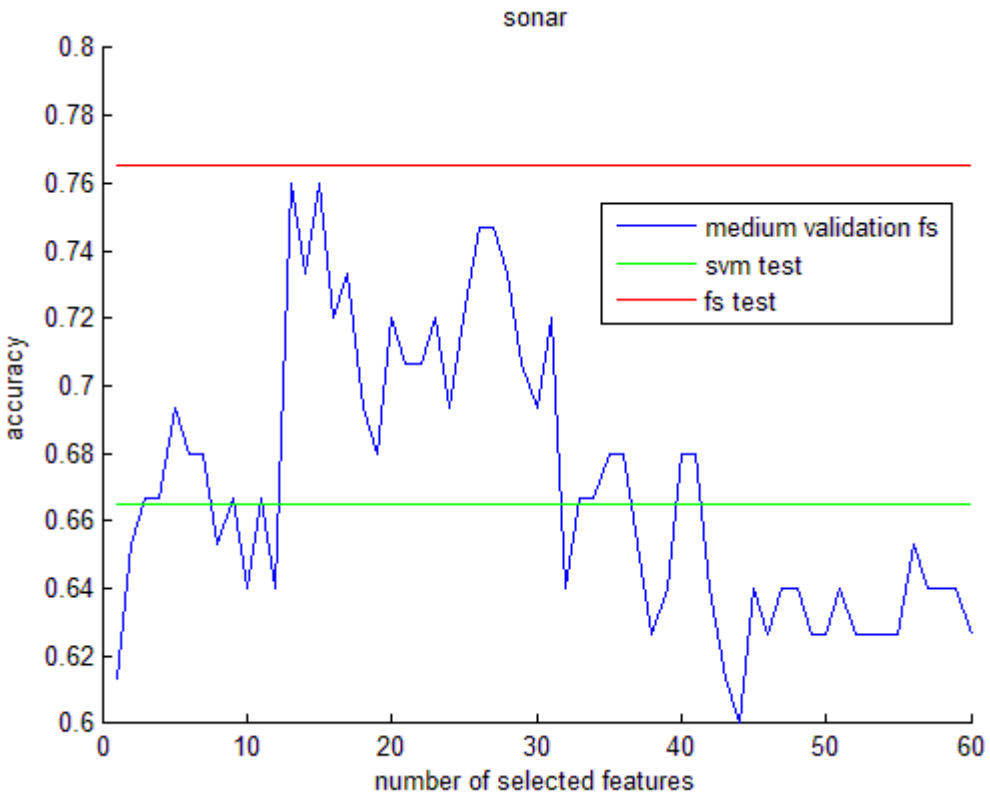

credit
accuracy
number of selected features
medium validation fs
svm test
fs test
0.85
0.8
0.75
0.7
0.65
0
5
10
15

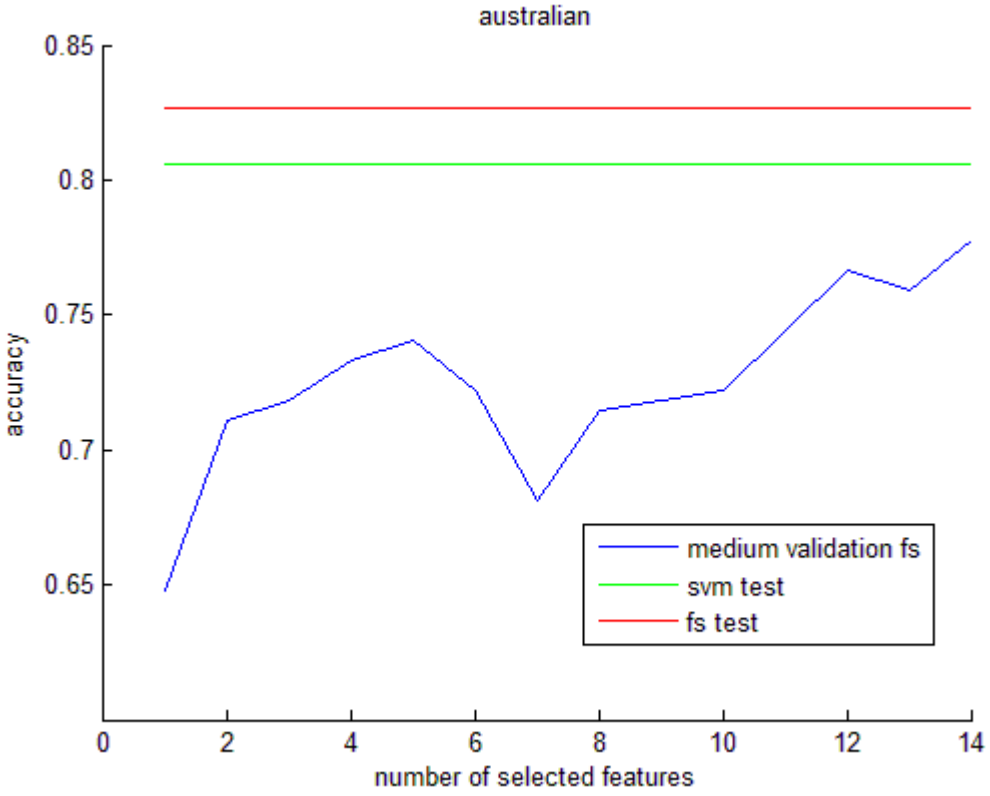

ionosphere
accuracy
number of selected features
medium validation fs
svm test
fs test
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0
5
10
15
20
25
30
35


pendigits
accuracy
number of selected features
medium validation fs
svm test
fs test
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8
10
12
14
16


pima
accuracy
number of selected features
medium validation fs
svm test
fs test
0.76
0.75
0.74
0.73
0.72
0.71
0.7
0.69
0.68
0.67
1
2
3
4
5
6
7
8


segmentation
accuracy
number of selected features
medium validation fs
svm test
fs test
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
2
4
6
8
10
12
14
16
18
20


sonar
accuracy
number of selected features
medium validation fs
svm test
fs test
0.8
0.78
0.76
0.74
0.72
0.7
0.68
0.66
0.64
0.62
0.6
0
10
20
30
40
50
60


australian
accuracy
number of selected features
medium validation fs
svm test
fs test
0.85
0.8
0.75
0.7
0.65
0
2
4
6
8
10
12
14

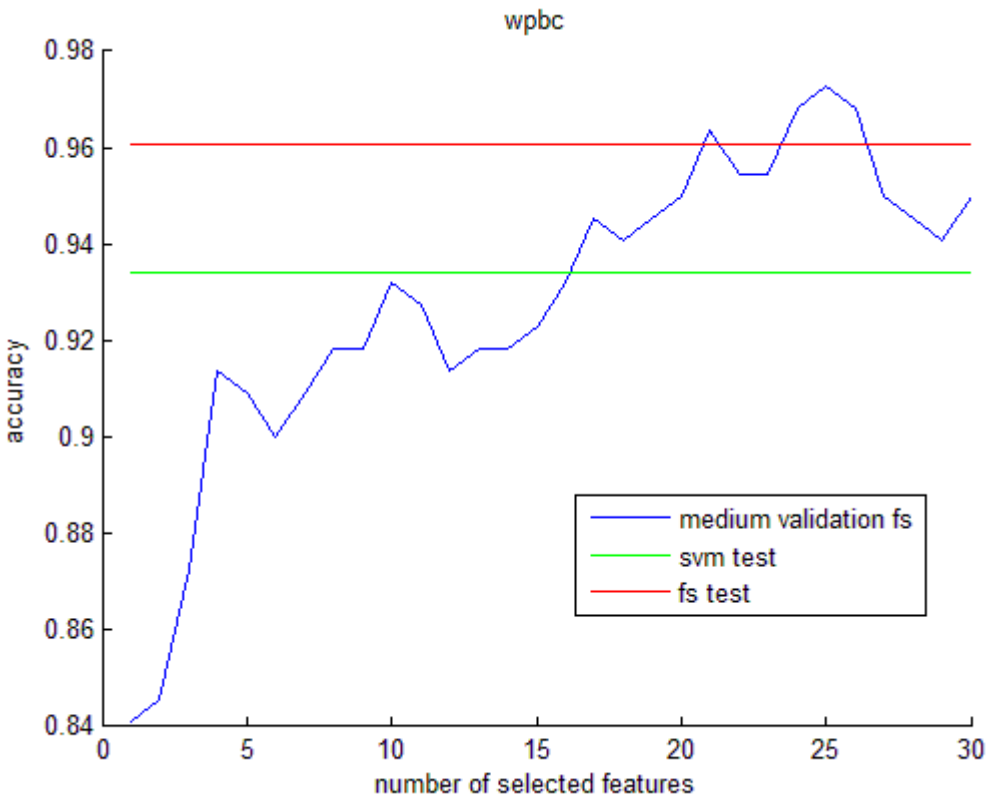


Figure 4.4 Graph of validation phases in SEFR algorithm.

Dataset name	Baseline SVM accuracy	SERF accuracy
Tictactoe	0,7042	**0,7842**
Vowel	0,5394	**0,9091**
Car	0,8827	**0,9848**
Credit	0,8031	**0,8450**
Ionosphere	0,6551	**0,8870**
Pendigits	0,9772	**0,9934**
Pima	0,7289	**0,7566**
Segmentation	0,9167	**0,9548**
Sonar	0,6650	**0,7650**
Australian	0,8058	**0,8263**
WPBC	0,9339	**0,9607**

Table 4.4 Results of SEFR approach classification.

Graphs in Figure 4.4 show how accuracy vary progressively expanding the feature subset, considering in each iteration an increased number of features in ranking order. Since a 5-time cross validation is applied, graph are the result of an average of the 5 iterations of the test.

Red line, indicated as *"fs test"* in the legend is the average (of the 5 cross validation) of the resulting tests on test set with the optimal features subset, determined in the validation phase.

Finally the green line is the baseline SVM resulting accuracy on test set. It is calculated using only labeled data in the learning phase.

It's important to notice that results of feature selection algorithm (*"fs test"*) are determined applying semi-supervised learning with the Self-training approach. For this reason, as you would expect, accuracy is usually significantly higher than the baseline reference.

conclusions

One of the first things that can be noticed from the validation graphs is that some of them follow an increasing trend with the enlargement of the selected features (in particular *pendigits*, *segmentation* and *vowels* dataset). As regards the datasets related to this kind of graph, it can be concluded that feature selection has not a real positive effect in the classification procedure, and the (eventually significant) accuracy increment in the final test on test set respect to the baseline SVM test is due to the semi-supervised classification. The advantages of semi-supervised learning are widely treated in machine learning literature, and are more evident when the difference between the size of labeled and unlabeled data is considerable, as in the protocol used in these tests.

Feature selection is more interesting in datasets like *sonar* or *ionosphere*. In these cases the graphs have a non-monotonic trend, with a peak in the corresponding optimal features subset. Then, adding features, the performance degrades. This can be considered an overfitting effect, in which too many features are not useful for the classification of data. In these cases feature selection is useful because permits to identify the most important features and to avoid overfitting. The datasets which present this problem have a large number of features (*sonar* has 60 features while *ionosphere* 36). Feature

selection represents an automatic way for the selection of useful features in datasets with the above described overfitting problem.

4.4.4 Test 4

Test phase 4 has consisted of test of DCPE approach [7] and clustering-based classification [9], after the application of the feature selection algorithm described in section 3.1.3. In this test, the used protocol is the same used for each experiment, as described in section 4.1. In the next the results will be presented in terms of accuracy of these two classification algorithms.

results of test 4

Dataset name	Baseline SVM accuracy	DCPE accuracy
Tictactoe	0.7116	**0.7189**
Vowel	0.4970	**0.4980**
Car	0.8886	0.8833
Credit	0.7969	**0.8016**
Ionosphere	0.6493	**0.7333**
Pendigits	0.9746	**0.9752**
Pima	0.6987	**0.7211**
Segmentation	0.9181	**0.9233**
Sonar	0.6800	0.6450
Australian	0.7942	**0.7985**
WPBC	0.9571	0.9304

Table 4.5 Results of DCPE with features selection.

Dataset name	Baseline SVM accuracy	Clustering-based accuracy
Tictactoe	0.7074	0.7042
Vowel	0.4879	0.4394
Car	0.8857	0.8483
Credit	0.7969	**0.8326**
Ionosphere	0.6957	**0.7855**
Pendigits	0.9694	0.9627
Pima	0.7224	**0.7579**
Segmentation	0.9200	0.8971
Sonar	0.6650	**0.7250**
Australian	0.8102	**0.8336**
WPBC	0.9357	**0.9393**

Table 4.6 Results of clustering-based classification with features selection.

conclusion

This experiment consists in the implementation of two classifiers which use semi-supervised techniques to classify test data. The classifiers are applied after a feature selection phase similar to the test 3 experiment. The feature selection, as can be expected, provides the best results in *ionosphere* and *sonar* datasets, as in test 3. For the other datasets, the performance is in many cases comparable with the baseline SVM approach.

As can be noticed, *ionosphere* dataset is the once with the better performance respect to the baseline, in both DCPE and clustering-based classification. This is due to the previous phase of feature selection; in fact, as seen in paragraph 4.3, feature selection drastically increases the accuracy in this dataset, which is otherwise subject to overfitting.

The clustering-based classifier has a good accuracy in *sonar* dataset. As said before, in this dataset the feature selection phase has a very good effect, due to the high number of features and the motivations explained in previous sections. So, good results could be expected from semi-supervised techniques. However, DCPE algorithm hasn't good performance on this dataset. This can be due to the different model of classification between DCPE and Self-training (used for the feature selection).

In other words, features' importance for the classification with Self-training is substantially different from the one related to the DCPE approach, for what regards the *sonar* dataset.

DCPE is an approach of semi-supervised learning based on diversity in the data and so it can provide good performance in these datasets which present a diversity in the wrong classified data. For example it has good performance in the *pima* dataset. This dataset presents an optimal features subset different from the total features set. The reason of the high accuracy result can also be due to the internal structure of the dataset, which can be more reliable to a diversity based approach. Moreover this result is similar to the result presented in [7] relatively to the *pima* dataset, which provides an error of 0.297 or 0.322 (depending on the used algorithm, if it is Nearest Neighbor or K-Nearest Neighbor). This means that feature selection has a limited effect on this dataset and the better performance respect to the baseline SVM classification is due to DCPE approach.

Clustering-based algorithm presents a more significant difference in accuracy performance. Using feature selection, this algorithm has a good accuracy in *ionosphere* and *sonar* datasets, as can be expected from the graphs of feature selection (Figure 4.4). It also has a higher accuracy in other datasets: *Credit*, *Pima* and *Australian*. This algorithm has instead a sensibly low accuracy respect to the baseline SVM in *Vowel*, *Car* and *Segmentation*. As can be observed, the clustering-based classification has more variable accuracy results respect to DCPE approach, which has usually similar performance to the baseline. This means that clustering is influenced by the dataset internal structure more deeply than the diversity-based algorithm.

In conclusion performance of clustering-based algorithm relies on the used dataset more than DCPE algorithm, so it can be necessary to previously test the performance of clustering and compare them with the baseline approach for every tested dataset.

4.4.5 Test 5

In test 5, DCPE and clustering-based classification are implemented using the SMCS classifier described in [6] as base classifier, instead of SVM. Moreover, a features selection phase is previously applied, as in test 4. In previous test, the base classifier used for clustering was a SVM with the radial basis function kernel: $e^{-(\gamma|u-v|^2)}$, with a gamma factor of 0.007 and a cost factor of 100, providing in output probability estimation. Expressed in the MATLAB syntax of libSVM, it is represented by the following option string:

' -g 0.07 -c 100 -t 2 -b 1'

In this test, instead, the baseline classifier is a classifier built as described in [6], with the generation of pseudo points and using the principles of single classifier based multiple classification scheme described in section 3.1.4.

For DCPE based approach, it is also used the SMCS algorithm instead of SVM, and adaboost algorithm as second classifier (as in test 4).

results of test 5

Dataset name	Baseline SVM accuracy	DCPE accuracy
Tictactoe	0.6705	**0.6768**
Vowel	0.4980	**0.5000**
Car	0.9037	0.8903
Credit	0.7829	0.7194
Ionosphere	0.6812	**0.7101**
Pendigits	0.9769	0.9625
Pima	0.6882	**0.6987**
Segmentation	0.9176	0.9100
Sonar	0.6450	0.6150
Australian	0.8131	0.7664
WPBC	0.9393	0.8821

Table 4.7 Results of DCPE using SMCS base classifier and features selection.

Dataset name	Baseline SVM accuracy	Clustering-based accuracy
Tictactoe	0.7011	0.6558
Vowel	0.4848	0.1101
Car	0.8862	0.6552
Credit	0.7953	0.5209
Ionosphere	0.6783	0.6174
Pendigits	0.9510	0.1268
Pima	0.7053	0.6474
Segmentation	0.9381	0.1500
Sonar	0.6600	0.4900
Australian	0.7883	0.6044
WPBC	0.9357	0.6518

Table 4.8 Results of clustering-based learner using SMCS base classifier and features selection.

conclusions

The semi-supervised algorithms DCPE tested using as base classifier the algorithm proposed in [6] provides different accuracy values respect to the test 4 (with SVM as base classifier).

DCPE classifier has good performance respect to the baseline test in *pima* an *ionosphere* datasets, like in the test 4. However the results are generally worse than in the test 4, where SVM is used as base classifier. From this test can be concluded that DCPE algorithm combined with SMCS approach [6] does not provide good performance, generally in every testes dataset.

Clustering-based classification has bad accuracy performance in each tested dataset. In every case the accuracy of clustering-based classifier is lower than the baseline. In some cases the accuracy is drastically reduced to the level of a random classifier. This is evident, for example, in the *vowel* dataset, where the high number of classes reduces considerably the accuracy. The accuracy has low levels in other datasets too; also in the ones with a good feature selection result, as *sonar* or *ionosphere*. In *segmentation* and *pendigits* datasets the accuracy has very low values. These values can be due to some problem of compatibility with the algorithms in the relative datasets. It is important to note that these datasets analysis is computationally more expensive than in the other cases. This is due to the

high number of samples and classes of *segmentation* and *pendigits*, combined with the complexity of the combination of different algorithms (SMCS, DCPE/clustering-based classification, feature selection).

In conclusion, it can be noticed that DCPE and clustering-based semi-supervised classification doesn't achieve good performance when combined with the classification algorithm proposed in [6]. This can be due to the relatively small amount of labeled data, that doesn't provide good performance in SMCS classification.

Chapter 5

Conclusions

As an overall conclusion, considering feature selection, we can notice that accuracy in every dataset obtained in test 3 is higher than the accuracy resulting from the other tested semi-supervised learning algorithms.

A reason for high accuracy results of test 3 respect to the other SSL algorithms, applied after a feature selection phase, can be the usage of Self-training as classification algorithm in test 3. Since feature selection is implemented following a Self-training strategy, it is expected to have good performance when classification is provided by Self-training algorithm. In other words we can say that the ranking of features provided by Algorithm 3.4 is strongly related to the Self-training classification, while, with other SSL algorithm, the optimal feature subset is not exactly the one provided by feature selection procedure described in paragraph 3.1.3. This results can be confirmed by good accuracy of DCPE or clustering-based algorithm tested in stand-alone mode, which is similar to the results provided in reference articles [7] and [9]. For what concerns clustering-based classification, must be noticed that protocol used for data analysis is different respect to the one used in the article, since in this thesis work, the protocol of data division proposed in [9] has been adopted for each test.

From the comparison of feature selection results and accuracy provided by methods in test 4, it can be concluded that the feature selection pattern described in section 3.1.3 is correlated to the SSL algorithm on which it is based (in this case the Self-training algorithm).

Finally, it can be concluded that features ranking algorithm described in [8] is a good instrument, useful to determine if the number of features is too high and might cause overfitting problem. However, as shown by results relative to *sonar* dataset in Table 4.4, it is not sure that feature selection would provide a better accuracy in datasets with a large number of features foe every classification algorithm.

Feature selection, normally, can sensibly increase accuracy in classification in high dimensional datasets (notice that the two SSL tested algorithms have high performance for what regards *ionosphere* and *sonar* datasets. Table 4.4 and 4.5). Despite that, it is useful to test the SSL algorithm in a stand-alone phase before to use feature selection. This is important because there could be problems of compatibility between the rank of features provided by the ranking algorithm and the Semi-supervised classifier used in the learning and test phases.

An alternative approach could consist in the implementation of an ad-hoc feature ranking algorithm, specifically based on the model of the used SSL classifier. For example in the case of DCPE, instead the features ranking calculated with the approach based on Self-training described in [8], optimal features subset can be determined with an feature ranking algorithm based on DCPE classifier. Therefore, feature selection with Self-training results to have the best performance in the analyzed datasets. However, in a generic dataset, the problems of Self-training described in the end of section 2.3.3 may occur. For example there could be problems of algorithm convergence, or early classification errors can propagate themselves and affect performance. Despite that, this methods was effective and has a quite low time consumption.

Another relevant conclusion is that, from executed tests, the best SSL method results to be DCPE classification with feature selection. This technique provides in many datasets better results than the baseline classifier (Table 4.5). A lower accuracy respect to base SVM is obtained in *sonar* and *WPBC*

datasets. For what regards *sonar* dataset, the possible causes of the low accuracy have been already discussed in section 4.4.4. From Table 4.6, it can be noticed that clustering based algorithm has a good accuracy for *sonar* dataset, sensibly higher than the baseline classifier.

The results of test 4 have been computationally expensive, especially the clustering based learner. This can be a drawback in many applications that need to dynamically classify data, such as analysis of web usage patterns or, in general, classification of data which change dynamically.

Finally we have taken into account the fusion of SSL techniques with SMCS approach, described in section 3.1.4. Results of this experiment are sensibly worse than previous tests, so it can be concluded that the usage of SVM as base classifier for DCPE and clustering based algorithms provides better performance than SMCS.

SMCS has good performance if tested in a stand-alone implementation. The fact that performance are lower if inserted in a more advanced SSL algorithm may be related to the presence of an additional classifier (the second classifier in DCPE co-training and Semi-Supervised Fuzzy c-mean clustering). In the algorithms iterations, the SMCS classifier uses the samples progressively labeled. For the structure of SMCS, which generate pseudo-points from existing samples, an initial error, caused by previous classification, can propagate itself in the pseudo-points generation phase and affect the overall accuracy. This effect is more evident in the clustering based results (Table 4.8), where the accuracy is considerably reduced respect to the baseline classifier. Finally, can be concluded that SMCS model doesn't provide good performance if combined with SSL techniques analyzed in this thesis work.

As a conclusion about the final results obtained in the tests, we can observe the good performance of feature selection and the importance of this method in order to avoid overfitting in some specific datasets (*ionosphere* and *sonar*). Moreover the DCPE and clustering based algorithm (section 3.1.1 and 3.1.2) have good performance if applied to the optimal feature subset obtained with the feature selection algorithm (section 3.1.3).

Bibliography

[1] - Introduction to data mining - pang-ning tan, Michael Steinbach, Vipin Kumar.

[2] – Semi-supervised Learning – Olivier Chapelle, Bernhard Schoolkopf, Alexander Zien.

[3] - Discriminative Models for Semi-Supervised Natural Language Learning - Sajib Dasgupta, Vincent Ng University of Texas at Dallas (2009).

[4] - Introduction to Machine Learning - Alex Smola and S.V.N. Vishwanathan.

[5] – A Survey of Decision Tree Classifier Methodology - S. Rasoul Safavian and David Landgrebe, IEEE Transactions on Systems, Man, and Cybernetics, (1991).

[6] - Single Classifier-based Multiple Classification Scheme for weak classifiers: An experimental comparison - Albert Hung-Ren Ko, Robert Sabourin (2013).

[7] - DCPE co-training for classification - Jin Xu, HaiboHe, HongMana (2010).

[8] - A semi-supervised feature ranking method with ensemble learning - Fazia Bellal, Haytham Elghazel, Alex Aussem (2011).

[9] - Using clustering analysis to improve semi-supervised classification - Haitao Gan, NongSang, RuiHuang, XiaojunTong, ZhipingDan (2011).

[10] – Introduction to Semi-Supervised Learning – Xiaojin Zhu, Andrew B.Goldberg (2009).

[11] - Computational Methods of Feature Selection – Huan Liu, Hiroshi Motoka – Chapman & all. / CRC.

[12] – libSVM documentation: http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf.

[13] - Introduction to Data Mining and its Applications - S. Sumathi, S.N. Sivanandam.

[14] - C-SVC (Boser et al., 1992; Cortes and Vapnik, 1995) – section 2.2.4

[15] – A. Blum, T. Mitchell - Combining labeled and unlabeled data with co-training, in: Proceedings of the 11th Annual Conference on Computational Learning Theory (1998).

[16] - Z.H. Zhou, M. Li - Tri-training: exploiting unlabeled data using three classifiers, IEEE Trans. Knowl. Data Eng. 17 (11) (2005).

[17] - M.M. Adankon, M. Cheriet, Help-training for semi-supervised support vector machines, Pattern Recognition 44 (9) (2011).

[18] - Shahshahani and D. Landgrebe - The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. (1994).

[19] – Bruce - Semi-supervised learning using prior probabilities and EM. In *IJCAI-01 Workshop on Text Learning: Beyond Supervision*, (2001).

[20] - Y. Grandvalet and Y. Bengio - Semi-supervised learning by entropy minimization. (2004).

[21] - Z.H. Zhou, M. Li, Semi-supervised learning by disagreement, (2010),

[22] - Dy, J., Brodley, C. Feature selection for unsupervised learning (2004).

[23] - Hopfield, J.J. -Artificial neural networks - Circuits and Devices Magazine, IEEE (1988).

[24] -Jie Cheng, Russell Greiner - Comparing Bayesian network classifiers – Proceeding UAI'99 Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence (1991).

[25] -Hearst, M.A., Dumais, S.T., Osman, E., Platt, J. – Support vector machines - Intelligent Systems and their Applications, IEEE (1998).

[26] – Yarowsky D. - Unsupervised word sense disambiguation rivaling supervised methods. - Proceedings of the 33rd annual meeting on Association for Computational Linguistics (1995).

[27] - Rilof E., Wiebe J., and Wilson T. - Learning subjective nouns using extraction pattern bootstrapping. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL (2003).

[28] - Maeireizo B., Litman D., and Hwa R. - Co-training for predicting emotions with spoken dialogue data. In Proceedings of the ACL 2004 on Interactive poster and demonstration sessions (2004).

[29] – Chuck Rosenberg, Martial Hebert, Henry Schneiderman, - Supervised Self-Training of Object Detection Models (2005).

[30] –K. Nigam, A. McCallum, S. Thrun and T. Mitchell – Text classification from labeled and unlabeled documents using EM – Machine Learning 39 (2000).

[31] – Datasets and information available in UCI websites: ***https://archive.ics.uci.edu/ml/machine-learning-databases***

[32] - Breiman, L. - Random forests. Machine Learning. 45 (2001).

[33] - Brown, G., Wyatt, J., Harris, R., & Yao, X. - Diversity creation methods: A survey and categorization - International Journal of Information Fusion, 6(1), 5–20 (2005).

[34] - Ko, A. H. R., Sabourin, R., & de Souza Britto, A. Jr. - Compound diversity functions for ensemble selection - International Journal of Pattern Recognition and Artificial Intelligence, 23(4), 659–686 (2009).

[35] - Kuncheva, L. I., & Whitaker, C. J. - Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy - Machine Learning (2003).

[36] - Kuncheva, L. I., Skurichina, M., & Duin, R. P. W. - An experimental study on diversity for bagging and boosting with linear classifiers. International Journal of Information Fusion, 3(2), 245–258 (2002).

Acknowledgements

I would like to express my special thanks to my supervisor professor Prof. Loris Nanni, who helps me in the code development and debugging phase and guides me during this thesis work. I would also like to express my gratitude to my family and friends, who support me during the thesis elaboration and my university studies.