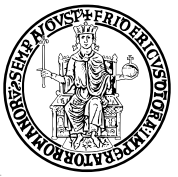


UNIVERSITÀ DEGLI STUDI DI NAPOLI
"FEDERICO II"

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

CORSO DI LAUREA IN INFORMATICA

STATE-OF-THE-ART CLUSTERING TECHNIQUES

Support Vector Methods and Minimum Bregman Information Principle

Tesi di Laurea in

INFORMATICA

Relatore

PROF.SSA Anna CORAZZA

Correlatore

PROF. Ezio CATANZARITI

Candidato

Vincenzo RUSSO

MATR. 50/550

Sessione Invernale

Anno Accademico 2006/07

A mia madre Elisa e a mio padre Giuseppe, per i sacrifici fatti e per i principi trasmessimi.

Alla mia famiglia tutta, per l'affetto, la fiducia e la pazienza.

Alla mia fidanzata Manuela, per il sincero appoggio e l'affettuosa pazienza.

Sommario

Il *clustering* è una metodologia di apprendimento automatico che ha come obiettivo la suddivisione di un insieme iniziale di oggetti in sottoinsiemi detti, appunto, *cluster*. La suddivisione avviene facendo sì che gli oggetti in un cluster siano quanto più simili tra di loro, mentre gli oggetti in cluster diversi risulteranno essere quanto più differenti tra di loro. Il clustering è una metodologia di apprendimento detta *non supervisionata*, poiché il raggruppamento degli oggetti in sottoinsiemi avviene in maniera totalmente automatica: l'informazione necessaria a tale raggruppamento è contenuta soltanto nei dati e nessun intervento esterno (umano) aggiunge informazione al fine di migliorare l'apprendimento. I campi applicativi del clustering sono molteplici. Un esempio è il clustering di documenti di testo: l'obiettivo in questo caso è quello di formare gruppi di documenti correlati tra loro, ovvero che trattino il medesimo argomento.
L'obiettivo del presente lavoro di tesi è stato quello di studiare e approfondire tecniche di clustering innovative e/o sperimentali. Le metodologie prese in esame sono due. La prima è conosciuta come *principio della Minimima Informazione di Bregman*. Tale principio generalizza il classico schema di relocazione adottato dal *K-means* a una vasta gamma di funzioni di divergenza, dette appunto divergenze di Bregman. Sulla base di tale principio è possibile definire un nuovo, più astratto, schema di clustering. In aggiunta, è stato formulato anche uno schema di *co-clustering* sulla base di tale principio, che, come osserveremo nel seguito, generalizza ulteriormente il processo di clustering.
Il secondo approccio studiato e approfondito è il *Support Vector Clustering*. Si tratta di un processo di clustering che fa uso di macchine di apprendimento allo stato dell'arte: le *Support Vector Machines*. Il Support Vector Clustering è attualmente oggetto di ricerca attiva, poiché trattasi ancora di un metodo sperimentale. Tale metodo è stato qui analizzato a fondo e sono stati apportati anche alcuni contributi, che permettono, ad esempio, di diminuire il numero di iterazioni, di aumentare l'accuratezza e di ridurre la complessità computazionale totale.
I domini applicativi principali sono stati il *text mining* e l'*astrophysics data mining*. Nell'ambito di questi domini applicativi ci si è mossi nella verifica e nell'approfondimento delle proprietà di entrambe le strategie summenzionate attraverso esperimenti mirati.
Tra i risultati emersi vi sono la robustezza rispetto alla presenza dei cosiddetti *missing values*, la riduzione della *dimensionalità*, la robustezza rispetto al rumore e ai cossiddetti *outliers*, la capacità di descrivere cluster di forma arbitraria.

Abstract

Clustering is an automatic learning technique aimed at grouping a set of objects into subsets or *clusters*. The goal is to create clusters that are coherent internally, but substantially different from each other. In plain words, objects in the same cluster should be as similar as possible, whereas objects in one cluster should be as dissimilar as possible from objects in the other clusters.
Clustering is an *unsupervised learning* technique, because it groups objects in clusters without any additional information: only the information provided by data is used and no human operation adds bits of information to improve the learning. The application domains are manifold. For example, the grouping of text documents: in this case the goal is the construction of groups of documents related to each other, i.e. documents treating the same argument.
The goal of this thesis is studying in depth state-of-the-art and experimental clustering techniques. We consider two techniques. The first is known as *Minimum Bregman Information principle*. Such a principle generalizes the classic relocation scheme adopted yet by K-means, in order to allow the employment of a rich gamma of divergence functions said just *Bregman divergences*. A new, more general, clustering scheme was developed on top of this principle. Moreover, a *co-clustering* scheme is formulated too. This leads to an important generalization, as we will see in the sequel.
The second approach is the *Support Vector Clustering*. It is a clustering process which relies on the state-of-the-art of the learning machines: the *Support Vector Machines*. The Support Vector Clustering is currently subject of active research, as it still is in early stage of development. We have accurately analyzed such a clustering method and we have also provided some contributions which allow a reduction in the number of iterations and in the computational complexity and a gain in accuracy.
The main application domains we have dealt to are the *text mining* and the *astrophysics data mining*. Within these application domains we have verified and accurately analyzed the properties of both methodologies, by means of dedicated experiments.
The results are given in terms of robustness w.r.t. the *missing values*, the dimensionality reduction, the robustness w.r.t. the noise and the outliers, the ability of describing clusters of arbitrary shapes.

ACKNOWLEDGMENTS

We meet a lot of people over the academic years. Whatever we say, each person plays a pretty much important role in our academic journey. Family, friends and darling believe in us and make the trip less hard. Smart colleagues that engage competitions and clever professors that stimulate and spur us, make us better students. Therefore, I wish to thank all these people.
Anyway, I feel the need to particularly thank some people. My honest (and equally distributed) thanks are for

- prof. Paola Festa, assistant professor of Operational Research and Mathematical Programming at University "Federico II" of Naples, for the support and the willingness in the early stage of the thesis and to let me know the fascinating world of machine learning under the supervision of my final advisor;
- prof. Giuseppe Longo, full professor of Astrophysics at University "Federico II" of Naples, for his enthusiastic interest in the work;
- prof. Stefano Russo, full professor of Computer Engineering at University "Federico II" of Naples and manager of the CINI I.Te.M. laboratories "Carlo Savy". He kindly allowed me to make use of the laboratories instruments and resources;
- ing. Alberto Raggioli, research and development manager at the M.E.T.A. s.r.l., for his real interest in the work.

CONTENTS

1 Introduction **1**
- 1.1 Clustering applications . . . 2
  - 1.1.1 Data reduction . . . 2
  - 1.1.2 Market research . . . 2
  - 1.1.3 Biology . . . 3
  - 1.1.4 Spatial Data Analysis . . . 3
  - 1.1.5 Astrophysics and astronomy . . . 3
  - 1.1.6 Information Retrieval . . . 4
- 1.2 Clustering process overview . . . 6
- 1.3 Data representation and notations . . . 7
  - 1.3.1 The vector space model . . . 7
  - 1.3.2 Objects similarity . . . 8
- 1.4 Conclusion . . . 8

2 Machine learning essentials **9**
- 2.1 Introduction . . . 10
  - 2.1.1 Machine learning and data mining . . . 10
- 2.2 Taxonomy of machine learning research . . . 10
  - 2.2.1 Learning paradigms . . . 11
- 2.3 Frameworks for learning machines . . . 12
  - 2.3.1 Linear and nonlinear separability . . . 13
- 2.4 Artificial Neural Networks . . . 14
- 2.5 Statistical machine learning . . . 15
  - 2.5.1 Background and general method . . . 15
- 2.6 Support Vector Machines . . . 16
  - 2.6.1 Kernels essentials . . . 17
  - 2.6.2 Maximal margin classifier . . . 19
  - 2.6.3 Linear Support Vector Machines . . . 21
  - 2.6.4 Nonlinear Support Vector Machines . . . 23
- 2.7 Conclusion . . . 24

3 Clustering and related issues **25**
- 3.1 Problem statement . . . 25

3.2 Taxonomy on clustering techniques . . . 26
3.2.1 Flat clustering and hierarchical clustering . . . 26
3.2.2 Partitional and non-partitional clustering . . . 28
3.2.3 Flat partitional clustering . . . 28
3.2.4 One-way clustering and multi-dimensional clustering . . . 29
3.2.5 Other clustering types . . . 30
3.3 Evaluation of clustering . . . 30
3.3.1 Internal criteria . . . 30
3.3.2 External criteria . . . 31
3.3.3 Relative criteria . . . 34
3.4 Estimating the number of clusters . . . 36
3.5 More on results assessment . . . 37
3.6 High dimensional data and noise . . . 37
3.6.1 Dimensionality reduction . . . 38
3.6.2 Subspace clustering . . . 39
3.7 Sparseness and missing values . . . 40
3.7.1 Types of missing data values . . . 40
3.7.2 Handling missing values . . . 41
3.8 Outliers handling . . . 43
3.9 Clusters of arbitrary shape . . . 44
3.10 Conclusion . . . 44

4 Previous works on clustering **45**
4.1 K-means . . . 46
4.1.1 Computational complexity . . . 47
4.1.2 Advantages and disadvantages . . . 48
4.1.3 K-means variations . . . 48
4.2 Expectation-Maximization . . . 48
4.2.1 Maximum Likelihood . . . 49
4.2.2 The algorithm . . . 50
4.2.3 Advantages and disadvantages . . . 51
4.3 Kernel Methods for Clustering . . . 52
4.3.1 Metric kernelization . . . 52
4.3.2 Advantages and disadvantages . . . 53
4.4 Conclusion . . . 53

5 Minimum Bregman Information principle for Co-clustering **55**
5.1 Towards the Bregman divergences . . . 56
5.1.1 Affine sets and relative interior . . . 56
5.1.2 Bregman divergences . . . 57
5.1.3 Bregman information . . . 58
5.2 Hard Clustering with Bregman divergences . . . 60
5.2.1 Bregman Hard Clustering algorithm . . . 63
5.3 Introduction to co-clustering problem . . . 64
5.4 The Bregman hard co-clustering framework . . . 65
5.4.1 Data matrices and random variables . . . 66

5.4.2 General problem statement . . . 66
5.4.3 Co-clustering bases . . . 67
5.4.4 Minimum Bregman Information . . . 69
5.4.5 Hard co-clustering problem formulation . . . 72
5.4.6 A meta algorithm . . . 73
5.5 Bregman Block Average Co-clustering . . . 77
5.5.1 Minimum Bregman Information . . . 77
5.5.2 Problem formulation . . . 78
5.5.3 Towards the algorithm . . . 78
5.5.4 Block average co-clustering as Matrix Factorization . . . 80
5.6 Computational complexity . . . 80
5.7 Obtaining previous works . . . 81
5.7.1 K-means . . . 81
5.7.2 Minimum Sum Squared Residue Co-clustering . . . 81
5.7.3 Information-Theoretic Co-clustering . . . 82
5.8 Novel applications . . . 83
5.8.1 Matrix approximation . . . 83
5.8.2 Missing values prediction . . . 83
5.9 Advantages . . . 84
5.9.1 Addressed issues . . . 85
5.10 Disadvantages . . . 86
5.10.1 Recently addressed issues . . . 87
5.11 Conclusion . . . 87

6 Support Vector Clustering **89**
6.1 Introduction to Support Vector Clustering . . . 90
6.1.1 Cluster description . . . 91
6.1.2 Cluster labeling . . . 95
6.1.3 Working with Bounded Support Vectors . . . 95
6.1.4 Execution scheme . . . 99
6.1.5 Complexity . . . 101
6.2 Issues and limitations . . . 102
6.3 Cluster labeling algorithms . . . 103
6.3.1 Support Vector Graph . . . 104
6.3.2 Proximity Graph . . . 105
6.3.3 Spectral Graph Partitioning . . . 108
6.3.4 Gradient Descent . . . 109
6.3.5 Cone Cluster Labeling . . . 117
6.3.6 Other works on Cluster Labeling . . . 123
6.3.7 Overall conclusion . . . 124
6.4 Kernel width exploration . . . 125
6.4.1 Secant-like kernel width generator . . . 125
6.4.2 Angle-decrement kernel width generator . . . 132
6.4.3 Other works on kernel width exploration . . . 134
6.5 Soft margin constant estimation . . . 134
6.6 Stopping criteria . . . 135

6.6.1 Stopping criteria based on number of clusters 135
6.6.2 Stopping criteria based on kernel width 136
6.6.3 Stopping criteria based on relative evaluation criteria 136
6.6.4 Conclusion 138
6.7 Mercer kernels for Support Vector Clustering 138
6.7.1 Basic kernel requirements 138
6.7.2 Normalized kernels properties 139
6.7.3 Support Vector Clustering and other kernels 140
6.7.4 Conclusion 141
6.8 Scaling down the overall time complexity 141
6.8.1 Improving the SVM training 142
6.8.2 Preprocessing data 143
6.9 Other works on Support Vector Clustering 143
6.10 Contribution summary 144
6.11 Conclusion 144
6.12 Future work 146

7 Alternative Support Vector Methods for Clustering 147
7.1 Kernel Grower 148
7.1.1 K-means alternative formulation 148
7.1.2 Kernel Grower formulation 149
7.1.3 Conclusion 150
7.2 Multi-sphere Support Vector Clustering 151
7.2.1 Process overview 151
7.2.2 Process in details 152
7.2.3 Conclusion 153
7.3 Overall conclusion 154

8 Support Vector Clustering software development 155
8.1 Software foundations 156
8.1.1 Software for the cluster description stage 156
8.1.2 Software for connected components 157
8.2 Class structure 157
8.2.1 Datasets 158
8.2.2 File loaders 159
8.3 Support Vector Clustering 159
8.4 Companion functions 160
8.5 Conclusion 161
8.6 Future work 161

9 Experiments 163
9.1 The experimental environment 164
9.1.1 Hardware and Operating System 164
9.1.2 The software 165
9.2 General setup of the algorithms 165
9.3 Evaluation and comparison of clustering results 166
9.3.1 Comparing results 166

9.4 Preliminary experiments: the data . . . 167
9.4.1 Iris Plant Database . . . 167
9.4.2 Wisconsin Breast Cancer Database . . . 167
9.4.3 Wine Recognition Database . . . 167
9.4.4 Quadruped Mammals . . . 168
9.4.5 Mushroom Database . . . 168
9.4.6 Internet advertisements . . . 169
9.5 Preliminary experiments: the results . . . 169
9.5.1 Iris Plant Database . . . 169
9.5.2 Wisconsin Breast Cancer Database . . . 173
9.5.3 Wine Recognition Database . . . 174
9.5.4 Quadruped Mammals . . . 177
9.5.5 Mushroom Database . . . 178
9.5.6 Internet advertisements . . . 180
9.5.7 Conclusion . . . 182
9.6 Outliers handling: the data . . . 183
9.6.1 Syndeca 02 . . . 184
9.6.2 Syndeca 03 . . . 184
9.7 Outliers handling: the results . . . 184
9.7.1 Support Vector Clustering . . . 185
9.7.2 Bregman Co-clustering . . . 185
9.7.3 Conclusion . . . 186
9.8 Missing Values in Astrophysics: the data . . . 188
9.8.1 Data details . . . 190
9.9 Missing Values in Astrophysics: the results . . . 191
9.9.1 Results on MVSG data . . . 192
9.9.2 Results on AMVSG data . . . 194
9.9.3 Conclusion . . . 195
9.10 Overlapping clusters in Astrophysics: the data . . . 197
9.11 Overlapping clusters in Astrophysics: the results . . . 198
9.11.1 Bregman Co-clustering . . . 199
9.11.2 Support Vector Clustering . . . 200
9.11.3 Conclusion . . . 200
9.12 Clustering in Text Mining: the data . . . 202
9.12.1 CLASSIC3 . . . 205
9.12.2 NewsGroup20 . . . 206
9.13 Clustering in Text Mining: the results . . . 207
9.13.1 CLASSIC3 . . . 207
9.13.2 SCI3 . . . 210
9.13.3 PORE . . . 211
9.13.4 Conclusion . . . 211
9.14 Conclusion . . . 214

10 Conclusion and Future Work **219**
10.1 MBI principle for Co-clustering 219
10.2 Support Vector Clustering 220
10.2.1 Contribution 221
10.3 Future Work 221
10.3.1 Connecting the two works 222
10.3.2 Improve and extend the SVC software 223

A One Class classification via Support Vector Machines **225**
A.1 The ν-Support Vector Machine 225
A.2 The One Class Support Vector Machine 226
A.3 SVDD and 1-SVM equivalence 227

B Resource usage of the algorithms **231**
B.1 Support Vector Clustering 231
B.2 Bregman Co-clustering 232

C Star/Galaxy separation via Support Vector Machines **237**
C.1 Training set and Test sets 237
C.2 The results 238
C.3 The software 238

D Thesis Web Log **239**
D.1 The Blog 240
D.1.1 The contents 240
D.2 The features 240
D.3 The technology 241

Bibliography **257**

LIST OF TABLES

5.1 Bregman divergences generated from some convex functions. . . . 58
5.2 Minimum Bregman Information (MBI) solution and optimal Lagrange multipliers for Squared Euclidean distance. 71
5.3 MBI solution and optimal Lagrange multipliers for I-divergence. . 72

6.1 Classification of point images in relation to the feature space hypersphere. 93

9.1 Summary of datasets used in the preliminary stage of experiments. 168
9.2 Bregman Co-clustering of the Iris Plant Database, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. The $\mathcal{C}_6$ scheme yields the best results both with Euclidean Co-clustering and Information-Theoretic Co-clustering. The best results are highlighted in bold. . 170
9.3 Bregman Co-clustering of the Iris Plant Database, with two feature clusters requested. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. The $\mathcal{C}_5$ scheme yields the best results both with Euclidean Co-clustering and Information-Theoretic Co-clustering. The best results are highlighted in bold. 171
9.4 Support Vector Clustering of the Iris Plant Database. The first table shows the Precision (P), Recall (R) and F1 for each class and for each Support Vector Clustering (SVC) instance. The second one shows the Accuracy and the Macroaveraging. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold. 172
9.5 Bregman Co-clustering of the Wisconsin Breast Cancer Database, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the overall Accuracy. The best results are highlighted in bold. 174

9.6 Bregman Co-clustering of the Wisconsin Breast Cancer Database, with three feature clusters requested. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the overall Accuracy. The best results are highlighted in bold. 175
9.7 Support Vector Clustering of the Wisconsin Breast Cancer Database. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold. 176
9.8 Bregman Co-clustering of the Wine Recognition Database, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. The best results are highlighted in bold. 177
9.9 Bregman Co-clustering of the Wine Recognition Database, with three feature clusters requested. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. The best results are highlighted in bold. 178
9.10 Support Vector Clustering of the Wine Recognition Database. The first table shows the Precision (P), Recall (R) and F1 for each class and for each SVC instance. The second one shows the Accuracy and the Macroaveraging. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold. 179
9.11 Bregman Co-clustering of the Quadruped Mammals, without feature clustering. The first table shows the Sensitivity (Sen.) and the Specificity (Spe.) for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. 180
9.12 Bregman Co-clustering of the Quadruped Mammals, with four features clusters. The first table shows the Sensitivity (Sen.) and the Specificity (Spe.) for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. 181
9.13 Support Vector Clustering of the Quadruped Mammals. The first table shows the Sensitivity (Sen.) and the Specificity (Spe.) for each class and for each co-clustering basis. The second one shows the Accuracy. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold. 182
9.14 Bregman Co-clustering of the Mushroom Database, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. 183

9.15 Bregman Co-clustering of the Mushroom Database, with three feature clusters requested. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. 184

9.16 Support Vector Clustering of the Mushroom Database. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold. 185

9.17 Bregman Co-clustering of the Internet advertisements, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. . 186

9.18 Bregman Co-clustering of the Internet advertisements, with three feature clusters requested. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. 186

9.19 Support Vector Clustering of the Internet advertisements. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold. 187

9.20 Summary of the preliminary stage results. Only the (best) accuracy reported. 187

9.21 Summary of datasets used in the outlier handling stage of experiments. 188

9.22 Support Vector Clustering and Bregman Co-clustering of *Syndeca 02* and *Syndeca 03* data. The first table shows the Accuracy and the Macroaveraging both for SVC and for Bregman Co-clustering. For the latter, only the best schemes are reported ($\mathcal{C}_1$ for the Euclidean co-clustering and $\mathcal{C}_6$ for the Information-Theoretic co-clustering). The second table is a legend that provides details about the SVC instances. 190

9.23 Summary of datasets used in the first session of the missing values clustering experiments. Both datasets are 25-dimensional and the actual attributes reporting missing values are the last 10. The data are divided in two classes: Star and Galaxy. 190

9.24 Summary of datasets used in the second session of the missing values clustering experiments. All datasets are 15-dimensional and the data are divided in two classes: Star and Galaxy. 191

9.25 Bregman Co-clustering of the MVSG5000 dataset, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. 192

9.26 Bregman Co-clustering of the MVSG5000 dataset, with three feature clusters. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. 193

9.27 Bregman Co-clustering of the MVSG10000 dataset, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. 193

9.28 Bregman Co-clustering of the MVSG10000 dataset, with three feature clusters. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. 194

9.29 Support Vector Clustering of the MVSG5000 data. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold. 195

9.30 Support Vector Clustering of the MVSG10000 data. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold. 196

9.37 Summary of datasets used in the Star/Galaxy unsupervised separation to verify the behavior with respect to strongly overlapping clusters. 196

9.31 Bregman Co-clustering of the AMVSG5000 dataset and related missing-valued variants, without feature clustering. The table shows the results obtained by the best co-clustering basis: the Completeness (Comp.), the Contamination (Cont.), and the Accuracy for each dataset. 197

9.32 Bregman Co-clustering of the AMVSG5000 dataset and related missing-valued variants, with three feature clusters. The table shows the results obtained by the best co-clustering basis: the Completeness (Comp.), the Contamination (Cont.), and the Accuracy for each dataset. 197

9.33 Bregman Co-clustering of the AMVSG10000 dataset and related missing-valued variants, without feature clustering. The table shows the results obtained by the best co-clustering basis: the Completeness (Comp.), the Contamination (Cont.), and the Accuracy for each dataset. 198

9.34 Bregman Co-clustering of the AMVSG10000 dataset and related missing-valued variants, with three feature clusters. The table shows the results obtained by the best co-clustering basis: the Completeness (Comp.), the Contamination (Cont.), and the Accuracy for each dataset. 198

9.35 Support Vector Clustering of the AMVSG5000 dataset and related missing-valued variants. The table shows the results obtained by the best co-clustering basis: the Completeness (Comp.), the Contamination (Cont.), and the Accuracy for each dataset. For the last dataset there are no results available due to an unexplained crash of the SVC software running on such data. 199

9.36 Support Vector Clustering of the AMVSG10000 dataset and related missing-valued variants. The table shows the results obtained by the best co-clustering basis: the Completeness (Comp.), the Contamination (Cont.), and the Accuracy for each dataset. For the last dataset there are no results available due to an unexplained crash of the SVC software running on such data. 200

9.38 Bregman Co-clustering of the Longo 01 data, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. . . . 201

9.39 Bregman Co-clustering of the Longo 01 data, with three feature clusters. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. . 201

9.47 Summary of datasets used in the text clustering experiments. . . . 201

9.40 Bregman Co-clustering of the Longo 02 data, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. . . . 202

9.41 Bregman Co-clustering of the Longo 02 data, with three feature clusters. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. . 202

9.42 Bregman Co-clustering of the Longo 03 data, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. . . . 203

9.43 Bregman Co-clustering of the Longo 03 data, with three feature clusters. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold. . 203
9.44 Support Vector Clustering of the Longo 01 data. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy and the details about the SVC instances. The best results are highlighted in bold. 204
9.45 Support Vector Clustering of the Longo 02 data. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy and the details about the SVC instances. The best results are highlighted in bold. 204
9.46 Support Vector Clustering of the Longo 03 data. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy and the details about the SVC instances. The best results are highlighted in bold. 205
9.48 Bregman Co-clustering of the CLASSIC3, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. The Euclidean $\mathcal{C}_6$ scheme yields the best results. 208
9.49 Bregman Co-clustering of the CLASSIC3, with 10 feature clusters. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. Best results are in bold. 209
9.50 Support Vector Clustering of the CLASSIC3. The first table shows the Precision (P), Recall (R), F1, Accuracy and Macroaveraging for each class and for each SVC instance. The second table shows the details about the SVC instances. In bold the best results. 210
9.51 Bregman Co-clustering of the SCI3, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. In bold the best results. 211
9.52 Bregman Co-clustering of the SCI3, with 20 feature clusters. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. In bold the best results. 212
9.53 Bregman Co-clustering of the PORE, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis (there are no results for Information-Theoretic instances because it did not separate the clusters). The second one shows the Accuracy and the Macroaveraging. In bold the best results. 213

9.54 Bregman Co-clustering of the PORE, with 20 feature clusters. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. In bold the best results. 214

B.1 The Support Vector Clustering (SVC) CPU usage. The time taken by an SVC instance is the sum of the time taken by every single iteration needed to find the most suitable kernel width out. The "n/a" wording indicates that an instance was not used for that experiment. For details about the various instances, refer to the tables in chapter 9. 232
B.2 The Euclidean Co-clustering (with feature clustering) CPU and Memory usage. The first is expressed in seconds, whereas the second one in bytes. The time taken by a co-clustering instance is the time of an iteration multiplied by 20. 233
B.3 The Information-Theoretic Co-clustering (with feature clustering) CPU and Memory usage. The first is expressed in seconds, whereas the second one in bytes. The time taken by a co-clustering instance is the time of an iteration multiplied by 20. 233
B.4 The Euclidean Co-clustering (without feature clustering) CPU and Memory usage. The first is expressed in seconds, whereas the second one in bytes. The time taken by a co-clustering instance is the time of an iteration multiplied by 20. 234
B.5 The Information-Theoretic Co-clustering (without feature clustering) CPU and Memory usage. The first is expressed in seconds, whereas the second one in bytes. The time taken by a co-clustering instance is the time of an iteration multiplied by 20. 235

C.1 Classification via Support Vector Machines (SVMs) of the Longo 01 data. The table shows the the Completeness and the Contamination for each class and for each Support Vector Machine (SVM) formalization, and also the overall accuracy for each SVM formalization. In bold the best results. 238
C.2 Classification via Support Vector Machines (SVMs) of the Longo 02 data. The table shows the the Completeness and the Contamination for each class and for each SVM formalization, and also the overall accuracy for each SVM formalization. In bold the best results. 238
C.3 Classification via Support Vector Machines (SVMs) of the Longo 03 data. The table shows the the Completeness and the Contamination for each class and for each SVM formalization, and also the overall accuracy for each SVM formalization. In bold the best results. 238

LIST OF FIGURES

1.1 An example of a dataset with a clear cluster structure (Manning et al., 2007, fig. 16.1). 2
1.2 Clustering of search results to improve user recall. None of the top hits cover the "Tiger Woods" sense of *tiger*, but users can easily access it by clicking on the "Tiger Woods" cluster in the *Clustered Results* panel provided by *Vivisimo* search engine. 4
1.3 A scheme of the clustering process. The dotted boxes indicate the pre-clustering steps; the dashed boxes indicate the post-clustering steps; the solid boxes represents the core of the clustering process. The clustering evaluation is marked as part of the core process because the internal clustering evaluation (see section 3.3) is part of the unsupervised learning procedure. 6

2.1 A 2D classification problem. We can see that there is an infinite number of decision surfaces that separate two linearly separable classes. (Manning et al., 2007, fig. 14.8). 13
2.2 A non linear problem. We can see that the linear separator (the dashed line) cannot put together all the objects represented by a fulfilled dot. (Manning et al., 2007, fig. 14.10). 14
2.3 A nonlinear separable problem in the data space $\mathcal{X}$ that becomes linear separable in the feature space $\mathcal{F}$. 19
2.4 Linear classification: definition of a unique hyperplane (the solid line) illustrated in a 2-dimensional data space. The (geometric) margin is the distance between the dashed lines. 20

3.1 A classic flat clustering results. 27
3.2 An illustration of typical hierarchical clustering results. 28
3.3 Purity is an external criterion for evaluation of cluster quality. Majority class and number of members of the majority class for the three clusters are: X, 5 (cluster 1); circle, 4 (cluster 2); diamond, 3 (cluster 3). Purity is $(1/17) * (5 + 4 + 3) \approx 0.71$ (Manning et al., 2007, fig. 16.4). 32
3.4 In this example we can see five clusters with a low level of outliers randomly distributed in data space. (Pei and Zaïane, 2006, fig. 13) . 42

3.5 An example of clusters with arbitrary shapes. (Pei and Zaïane, 2006, fig. 11) 43

5.1 Schematic diagram of the six co-clustering bases. In each case, the summary statistics used for reconstruction are the expectations taken over the corresponding dotted regions. The matrices are "reordered", i.e. the rows and the columns are arranged according to cluster order to reveal the various co-clusters. (Banerjee et al., 2007, fig. 1) 69

5.2 **(a)**: Co-clustering-based approximation of a simple 50×50 synthetic matrix (M_2) using various co-clustering bases, squared distortion and $k = l = 5$. While the matrix is too complex for $\mathcal{C}_1$, all bases from $\mathcal{C}_2$ onwards get an accurate approximation. **(b)**: Co-clustering-based approximation of a simple 50×50 synthetic matrix (M_6) using various co-clustering bases, squared distortion and $k = l = 5$. In this case the original matrix is fairly complex and only $\mathcal{C}_6$ gets an accurate approximation. All other schemes have more errors, with the simple bases ($\mathcal{C}_1$ and $\mathcal{C}_2$) having high errors. Note that all matrices are shown with a consistent permutation (which the co-clustering finds) for easy visual comparison (Banerjee et al., 2007, fig. 3, 4). 84

6.1 A graphical example of a data mapping from input space to feature space using the Minimum Enclosing Ball (MEB) formulation. . . . 91

6.2 Clustering of a dataset containing 183 points, with $C = 1$. Support vectors are designated by small circles. The figures from (a) to (d) represents the clustering executed at increasingly greater kernel width values (Ben-Hur et al., 2001, fig. 1). 96

6.3 Clustering with and without Bounded Support Vectors (BSVs). The inner cluster is composed of 50 points generated from a Gaussian distribution. The two concentric rings contains 150/300 points, generated form a uniform angular distribution and a radial Gaussian distribution. (a) The rings cannot be distinguished when $C = 1$. Only the inner cluster was separated. (b) Outliers allow easy clustering and all clusters was separated (Ben-Hur et al., 2001, fig. 3). 97

6.4 Clusters with overlapping density functions require the introduction of BSVs (Ben-Hur et al., 2001, fig. 4). 98

6.5 In the case of significant overlap between clusters the SVC algorithm identifies clusters according to dense cores, or maxima of the underlying probability distribution. (Ben-Hur et al., 2001, fig. 5). . 99

6.6 A false positive situation: all sample points on $\overline{tt'}$ are inside the minimal hypersphere although the two points are in different clusters (Lee and Daniels, 2005b, p. 69). 104

6.7 A false negative situation: all sample points on $\overline{tt'}$ are outside the minimal hypersphere although the two points are in the same cluster (Lee and Daniels, 2005b, p. 69). 104

6.8 The shaded regions represent clusters of $L_r(\overline{r})$ whose boundary is denoted by solid lines. The fulfilled dots are the Stable Equilibrium Points (SEPs), while the plus signs are the transition points (Lee and Lee, 2006, fig. 1a). 113

6.9 Developing an approximate cover of a part of P, where $\vec{v}_i$ and $\vec{v}_j$ are Support Vectors (SVs) and $\phi(\vec{v}_i)$ and $\phi(\vec{v}_j)$ are their feature space images, respectively. (a) $\Theta = \angle(\phi(\vec{v}_i)O\vec{a}) = \angle(\phi(\vec{v}_j)O\vec{a})$; (b) $\vec{a'}$ is the intersection of $\vec{a}$ with the unit ball surface; (c) the length $\|\vec{a}\|$ is $\langle\phi(\vec{v}_i), \vec{a'}\rangle$; (d) $\Delta = \langle\phi(\vec{v}_i), \vec{a}\rangle = 1 - R^2 = \|\vec{a}\|^2$. Note that this is only a 2D illustration; the actual feature space is high-dimensional (Lee and Daniels, 2006, fig. 2). 119

6.10 Support vector cones projections into data space, i.e. support vector spheres. They are represented by circles with the same radius. The outermost contours are the real contours of the clusters (Lee and Daniels, 2005b, fig.16). 120

6.11 Contour ϵ-expansion to face the weak connectivity. With the original contour the two points are detected as disconnected with the path sampling technique. The same two points result connected if we ε-expand the contour (Lee and Daniels, 2005b, fig.14). 124

6.12 An example of how the secant-like kernel width generator works. The curve through all fulfilled black dots is the approximation of the R^2 curve. The sequence of q values is $q_1, q_2, q_3, q_4, \cdots$. In the figure, N is the size of the input dataset (Lee and Daniels, 2005a, fig.1). 129

6.13 An example of how the angle decrement kernel width generator works. The angle is decreased at each step. The sequence of q values is $q_1, q_2, q_3, \cdots$. In the figure, n is the size of the input dataset (readapted from Hartono and Widyanto (2007, fig.4)). 133

8.1 The first three basic classes of the *damina* library. They are all interfaces which establish the mandatory methods for all implementing classes. 156

8.2 The *AbstractSVM* class inherits from the *ClassificationEngine* and provides all the basic methods for all SVMs. The *OneClassSVM* is not a direct child of the *AbstractSVM*: it inherits from the *NuSVM*, which implements the ν-SVM formalism. The *CSVM* implements the classic Vapnik SVM formulation. 157

8.3 A *Dataset* is a collection of *Point* which, in turn, is a collection of *Feature*. 158

8.4 The *FileLoader* interface and the *LibsvmFileLoader* class. 159

8.5 The *SVClustering* inherits from *ClusteringEngine* and implements the original Support Vector Clustering (SVC) algorithm. It uses also the *OneClassSVM* class and the *Graph* one from the Boost library. The *CCLSVClustering* inherits from *SVClustering* and overrides the cluster labeling algorithm. 160

9.1 The *Syndeca 02* 3D plot was made by means of the first three attributes. Since we asked the SynDECA generator to produce a fairly separable problem, we were sure that the plot would be faithful with any feature. Since the features are unnamed, in the plot we wrote "col1", "col2", and "col3" for the first, the second, and the third feature respectively. The outliers are represented by empty circles. 188

9.2 Since the dataset is 3-dimensional, this plot exactly represents *Syndeca 03* data. The features are unnamed, so in the plot we have written "col1", "col2", and "col3" for the first, the second, and the third feature respectively. The outliers are represented by empty circles. 189

10.1 Examples of Bregman balls. The two ones on the left are balls obtained by means of the *Itakura-Saito* distance. The middle one is a classic *Euclidean* ball. The other two are obtained by employing the *Kullback-Leibler* distance (Nock and Nielsen, 2005, fig. 2). 223

A.1 Data descriptions by the 1-SVM and the Support Vector Domain Description (SVDD) where the data is normalized to unit norm (Tax, 2001, fig. 2.11). 227

D.1 A screenshot of the Thesis Web Log. 240

LIST OF ALGORITHMS

1 The K-means algorithm 49
2 Bregman hard clustering algorithm 62
3 Bregman generalized hard co-clustering algorithm 76
4 Bregman block average co-clustering algorithm 79
5 The general execution scheme for the SVC 101
6 Complete Graph Cluster Labeling 105
7 Support Vector Graph Cluster Labeling 106
8 Proximty Graph Cluster Labeling 107
9 Find the SEPs for Gradient Descent Cluster Labeling 114
10 Gradient Descent Cluster Labeling - Original Version 115
11 Gradient Descent Cluster Labeling - Enhanced Version 116
12 Cone Cluster Labeling 122
13 Cone Cluster Labeling - Version with our own contribution 123
14 Secant-like kernel width generator 128
15 The Kernel Grower (KG) algorithm 150
16 The Multi-sphere Support Vector Clustering (MSVC) algorithm . . 151

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
AMVSG	Artificial Missing-Valued Stars and Galaxies
ANNs	Artificial Neural Networks
BASIC	Beginner's All purpose Symbolic Instruction Code
BC	Bâdoiu-Clarkson
BI	Bregman Information
BVD	Block Value Decomposition
BSV	Bounded Support Vector
BSVs	Bounded Support Vectors
CALTECH	California Institute of Technology
CCL	Cone Cluster Labeling
CG	Complete Graph
CINI	Consorzio Interuniversitario Nazionale per l'Informatica
CPU	Central Processing Unit
CURE	Clustering Using REpresentatives
CVM	Core Vector Machine
CVMs	Core Vector Machines
DB	Davies-Bouldin
DBSCAN	Density Based Spatial Clustering of Applications with Noise
DD	Delaunay Diagrams
DF	Document Frequency

DNA DeoxyriboNucleic Acid

DR Data Release

EM Expectation-Maximization

ERM Empirical Risk Minimization

ESTs Expressed Sequence Tags

FN False Negative

FP False Positive

GD Gradient Descent

GK Goodman-Kruskal

GIS Geographical Information Systems

HTML Hyper Text Mark-Up Language

IB Information Bottleneck

IDF Inverse Document Frequency

i.i.d. Independent and identically-distributed

I.Te.M. Informatica e Telematica Multimediali

KG Kernel Grower

KL Kullback-Leibler

KKT Karush-Kuhn-Tucker

k-NN k-Nearest Neighbors

LS-SVM Least Squares Support Vector Machines

LBG Linde-Buzo-Gray

MAR Missing At Random

MBI Minimum Bregman Information

MCAR Missing Completely At Random

MDS Multidimensional Scaling

MEB Minimum Enclosing Ball

MEBB Minimum Enclosing Bregman Ball

MFA Mixture of Factors Analyzers

MLP Multi Layer Perceptron

MNAR Missing Not At Random

MST Minimum Spanning Trees

MSVC Multi-sphere Support Vector Clustering

MVSG Missing-Valued Stars and Galaxies

NG Neural Gas

NG20 NewsGroup20

NLP Natural Language Processing

NMI Normalized Mutual Information

NN Nearest Neighbor

NP Non-deterministic Polynomial time

OCR Optical Character Recognition

OOP Object Oriented Programming

OPTICS Ordering Points To Identify the Clustering Structure

P Precision

PCA Principal Component Analysis

PG Proximity Graph

QP Quadratic Programming

R Recall

RBF Radial Basis Function

RI Rand Index

RSS Residual Sum of Squares

RVM Relevance Vector Machine

SDSS Sloan Digital Sky Survey

SEP Stable Equilibrium Point

SEPs Stable Equilibrium Points

SGP Spectral Graph Partitioning

SMO Sequential Minimal Optmization

SOMs Self-Organizing Maps

SOR Successive Over-relaxation

SV Support Vector

SVs Support Vectors

SVC Support Vector Clustering

SVD Singular Value Decomposition

SVDD Support Vector Domain Description

SVG Support Vector Graph

SVM Support Vector Machine

SVMs Support Vector Machines

URL Uniform Resource Locator

UEP Unstable Equilibrium Point

TF Term Frequency

TN True Negative

TP True Positive

TRN Topology Representing Networks

VC Vapnik-Chervonenkis

CHAPTER

ONE

Introduction

«Number rules the universe.»

PYTHAGORAS
(MACHALE, 1993)

CLUSTERING is a technique to group a set of objects into subsets or *clusters* (Berkhin, 2002). Usually objects are described by attributes, also called *features*. The goal is to create clusters that are coherent internally, but substantially different from each other, on a per-feature basis. In plain words, objects in the same cluster should be as similar as possible, whereas objects in one cluster should be as dissimilar as possible from objects in the other clusters.
From a *machine learning*[1] perspective clusters correspond to *hidden patterns* and clustering is the most common form of *unsupervised learning*. No supervision means that there are no human experts who have assigned objects to classes. In clustering, it is the distribution and the nature of data that will determine cluster membership, in opposition to the classification (see subsubsection 2.2.1.1) where the classifier learns the association between objects and classes from a so called *training set*, i.e. a set of data correctly labeled by hand, and then replicates the learnt behavior on unlabeled data.
From a practical perspective clustering plays an outstanding role in *data mining*[2] applications such as scientific data exploration, information retrieval and text mining, web analysis, bioinformatics, and many others.
This thesis focus on *Clustering* and aims at introducing state-of-the-art clustering techniques, with experiments across multiple application domains and some contribution to the enhancement of techniques and/or their implementations.

[1] Clustering is the subject of active research in several fields, such as statistics, pattern recognition, machine learning, data mining, etc.

[2] More details on machine learning and data mining in chapter 2.

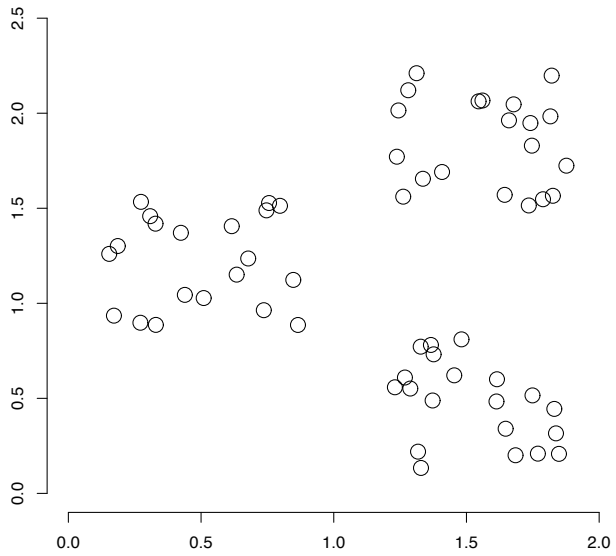


Figure 1.1: *An example of a dataset with a clear cluster structure (Manning et al., 2007, fig. 16.1).*

1.1 Clustering applications

Clustering is the most common form of unsupervised learning and is a major tool in a number of applications in many fields of business and science. Hereby, we summarize the basic directions in which clustering is used.

1.1.1 Data reduction

Cluster analysis can contribute in compressing the information included in data. In several cases, the amount of available data is very large and its processing becomes computationally very demanding. Clustering can be used to partition data sets into a number of interesting clusters. Then, instead of processing the data set as an entity, we adopt the representatives of the defined clusters in our process. In this way, data compression is achieved.

1.1.2 Market research

Cluster analysis is widely used in market research when working with multivariate data from surveys and test panels. Market researchers use cluster analysis to partition the general population of consumers into market segments and to better understand the relationships between different groups of consumers/potential customers. Also, it could be used to understand what kind of new product is a good investment.

1.1.3 Biology

In biology clustering has many applications. In the fields of plant and animal ecology, clustering is used to describe and to make spatial and temporal comparisons of communities (assemblages) of organisms in heterogeneous environments; it is also used in plant systematics to generate artificial phylogenies or clusters of organisms (individuals) at the species, genus or higher level that share a number of attributes.
In computational biology and bioinformatics clustering is used to build groups of genes with related expression patterns. Often such groups contain functionally related proteins, such as enzymes for a specific pathway, or genes that are co-regulated. High throughput experiments using Expressed Sequence Tags (ESTs) or DNA microarrays can be a powerful tool for genome annotation, a general aspect of genomics. Clustering can also be used to group homologous sequences into gene families. This is a very important concept in bioinformatics, and evolutionary biology in general.

1.1.4 Spatial Data Analysis

Due to the huge amounts of spatial data that may be obtained from Geographical Information Systems (GIS), satellite images, medical equipment, image database exploration, and many others, it is expensive and difficult for the users to examine spatial data in detail. Clustering may help to automate the process of analysing and understanding spatial data. It is used to identify and extract interesting characteristics and patterns that may exist in large spatial databases.

1.1.4.1 Image and video analysis

Managing increasingly large image repositories is a technically challenging problem, which in recent years has attracted extensive research interests. Data clustering is a standard technique that is also useful for image database management. To cluster images into visually homogeneous groups, images are represented by their visual features such as colour histogram and texture descriptors. A challenging task in this application area is to identify which features are more important to a given cluster. For instance, when a colour histogram is used to represent images for clustering, information about which colour bins are more strongly associated with a particular cluster is not known.

1.1.5 Astrophysics and astronomy

Clustering in astrophysics and astronomy is used to detect groups of stars, galaxies, asteroids and other objects in the space. The amount of data in these application fields is very huge and unsupervised learning techniques play a crucial role in the mining of the astronomical patterns. Usually the objects are represented in the vector model (see section 1.3) and the features are values from experimental measurements through sophisticated instruments.

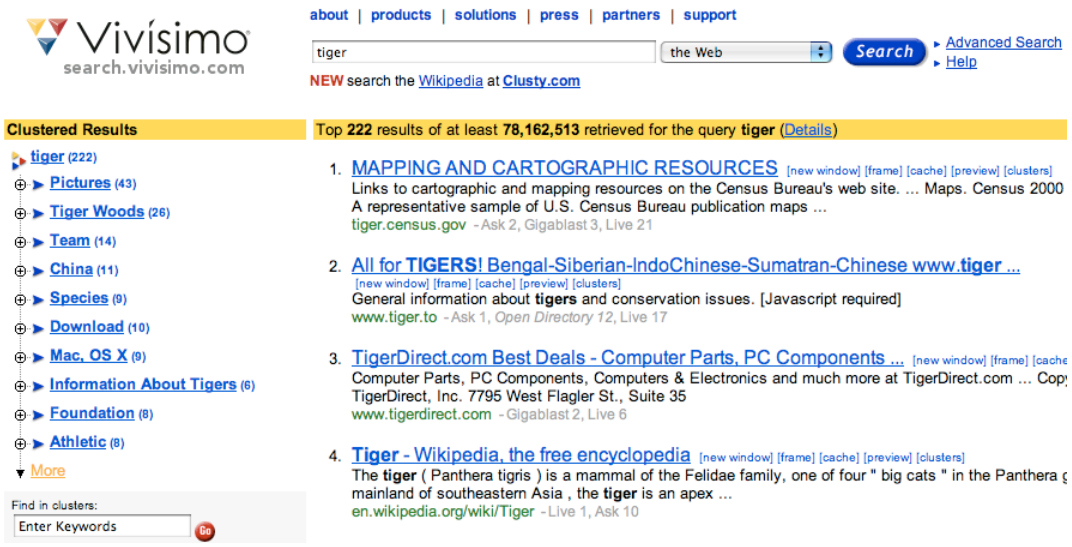


Figure 1.2: *Clustering of search results to improve user recall. None of the top hits cover the "Tiger Woods" sense of* tiger*, but users can easily access it by clicking on the "Tiger Woods" cluster in the* Clustered Results *panel provided by* Vivisimo *search engine.*

The distinctive characteristic of this domain application is the hugeness of the datasets. Usually real problems are represented by datasets with millions objects. Another peculiarity is the frequent presence of missing values due to the fail of a measurement, i.e. a feature does not have any value for a particular object because the measurement failed.

1.1.6 Information Retrieval

The *cluster hypothesis* states the fundamental assumption we make when using clustering in *information retrieval*.[3]

> **Cluster hypothesis.** Documents in the same cluster behave similarly with respect to relevance to information needs.

The hypothesis states that if there is a document from a cluster that is relevant to search request, then it is likely that other documents from the same cluster are also relevant. This is because the clustering puts together documents that share many words (the features in document clustering). There are different applications of clustering in information retrieval. They differ in the set of documents that they cluster (collection, result sets, etc.) and the aspect of an information retrieval system that they try to improve (effectiveness, accuracy, user experience, etc.). But they are all based on the aforementioned cluster hypothesis (Manning et al., 2007).

[3]The *cluster hypothesis* holds in case the documents are encoded as *bag of words*, i.e. by using the *vector space model*. See subsection 1.3.1 for details about the vector space model.

The first application is the *result set clustering*. In this case the goal is to improve the user experience by grouping together similar search results. Usually it is easier to scan few coherent groups rather than many individual documents, especially if the search terms have different meanings. For instance, let us consider the search term *tiger*; three frequent senses refer to the animal, the golf player "Tiger Woods" and a version of the Apple operating system. If the search engine presents the results showing a number of clusters like "animals", "Tiger Woods", "Mac", and so on, it will allow the end-user to easily discard the results that are not relevant to his search query (see Figure 1.2).[4]
Another application of clustering in information retrieval is the *scatter-gather* interface. Also in this case, the goal is to improve the *user experience*. Scatter-gather clusters the whole collection to get groups of documents that the user can select ("gather"). The selected groups are merged and the resulting set is again clustered. This process is repeated until a cluster of interest is found.
In the *web mining*, we can find several applications of the clustering in information retrieval. The web mining is a special case of the text documents mining. Currently the web applications that perform the so-called *knowledge aggregation* are proliferating on Internet. Such web softwares heavily rely on data mining techniques, especially clustering. Anyway, in most of these cases the clustering is semi-supervised because the Web are moving towards a more "semantic" conception. With the term "semi-supervised" we mean that the content have some representative attributes which depends on human operations.[5]
Clustering is also used to increase the precision and/or the recall. The standard inverted index is used to identify an initial set of documents that match the query, but then other documents from the same clusters is added even if they have low similarity to the query. For example, if the query is "car" and several car documents are taken from a cluster of automobile documents, then we can add documents from this cluster that use terms other than "car" (e.g. "automobile", "vehicle", etc.). This can increase recall since a group of documents with high mutual similarity is often relevant as a whole.
This idea has been also used for *language modeling*. It has been showed that to avoid sparse data problems in the language modeling approach to the information retrieval, the model of a document d can be interpolated with a collection model (Manning et al., 2007, chap. 12). But the collection contains many documents with words untypical of d. By replacing the collection model with a model derived from the document cluster, we get more accurate estimates of the occurrence probabilities of words in the document d.
Finally, the clustering was recently used also for address the missing values issue:[6] each document usually contains only a small number of the words chosen for representing the documents in a set. This makes the data matrix (see section 1.3) very sparse. So, the clustering is used to cluster also the words with respect to the documents.

[4]To see such a search engine in action, visit `http://vivisimo.com/` or `http://clusty.com`.

[5]For instance, tagging a content by means of keywords.

[6]See section 3.7.

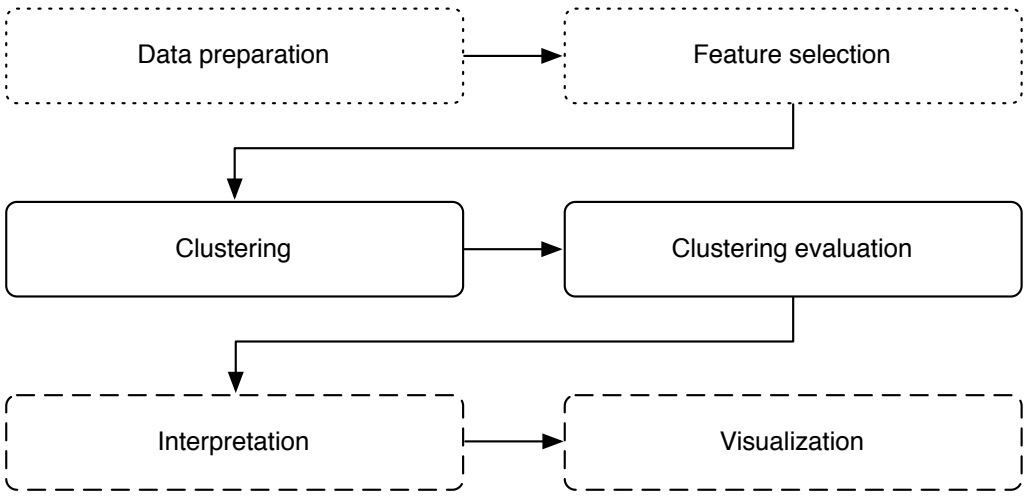


Figure 1.3: *A scheme of the clustering process. The dotted boxes indicate the pre-clustering steps; the dashed boxes indicate the post-clustering steps; the solid boxes represents the core of the clustering process. The clustering evaluation is marked as part of the core process because the internal clustering evaluation (see section 3.3) is part of the unsupervised learning procedure.*

The words clustering is also used for dimensionality reductions, since another distinctive characteristic of the text data are the huge amount of features which describe an object, i.e. the number of words are usually very large (thousands or even millions).

1.2 Clustering process overview

The whole clustering process does not consist only of the unsupervised learning procedure, but there are other additional steps. Before running the learning procedure, we could need to prepare data in the format handled by the clustering algorithm implementation or perform a *feature selection* process to select the more appropriate features from a list of many attributes.[7] After data preparation, we are ready to run the pure clustering algorithm. Successively we need to evaluate the clustering results, to understand how good the results are or whether the choice of the number of requested clusters is correct. Eventually, we have the final clustering and the last steps are the interpretation and the visualization of results. The pre-clustering and post-clustering steps are usually optional.
Anyway, whatever of the above steps we choose to employ, the ultimate goal of clustering is to assign points of a dataset to a finite system of k subsets, namely *clusters*, using some *similarity measure* to understand which points are in the same cluster. In the most intuitive case we want to find non-overlapping clusters, i.e. clusters are required not to intersect and their union must be equal to the original

[7] In several cases we do not use all available attributes in order to limit the dimensionality (see section 3.6).

dataset. So, let $\mathcal{C}_i$ be the i-th subset of the dataset $\mathcal{X}$ such that $\mathcal{C}_i$ is a cluster and let k be the number of desired clusters, we have

$$\mathcal{X} = \bigcup_{i=1}^{k} \mathcal{C}_i \tag{1.1}$$

and

$$\bigcap_{i=1}^{k} \mathcal{C}_i = \oslash \tag{1.2}$$

However, this is only the most general problem formulation and at the same time it is the simplest one. In many other situations we would admit overlapping clusters, a hierarchical structure of clusters or both of them; furthermore we could find in datasets several points that we call *outliers*, which do not belong to any of possible clusters. In the presence of outliers the union of the clusters could not be equal to the original dataset, depending on how we choose to deal with these "foreign" points.

1.3 Data representation and notations

In this section we introduce the data representation assumed[8] in the rest of this thesis and fix our notations to avoid misunderstanding, since there are several equivalent terminologies to refer to the same things.

1.3.1 The vector space model

The representation of a set of objects as vectors in a common vector space is known as the *vector space model* and is fundamental for a variety of machine learning and data mining applications ranging from regression, classification, clustering and others.
Let $o_i \in \mathcal{D}$ be the i-th object of a dataset $\mathcal{D}$, we denote with $\vec{x}_i$ the vector derived from the object o_i, where $\vec{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$, i.e. the set $\mathcal{X}$ usually is a linear subspace of $\mathbb{R}^d$. Each vector component is called *feature*[9] or *attribute*.
With a slight abuse of notation, we call the set $\mathcal{X}$ the *dataset*, notwithstanding the real dataset is the set of objects $\mathcal{D}$, whereas the elements of $\mathcal{X}$ are usually called *points*[10]. Hence we have

$$\mathcal{X} = \{\vec{x}_i | \vec{x}_i = (f_{i_j})_{j=1..d}\}_{i=1..n}$$

[8] Unless differently specified.

[9] It actually is an abuse of notation, because the vector component is an *instance* of a feature which is often represented by an *encoding*. For instance, a chair is an object represented by a variety of features, such as the number of legs, the color, the material etc. In a vector space model, a chair will be represented by a real-valued vector whose components will be an encoding of each feature instance.

[10] Synonymously, *objects*, *instances*, *patterns*, etc.

where n is the cardinality of the set $\mathcal{X}$, $\vec{x}_i$ are the points, d is the dimensionality of the linear subspace and f_{i_j} are the features, where $j = 1, 2, .., d$.
Such point-by-feature data format conceptually corresponds to an $n \times d$ data matrix used by the majority of the clustering algorithms; each matrix row represents an object, whereas in each matrix column we find the value of a specific feature for each objects.

1.3.2 Objects similarity

Once the data are represented in a vector space model, we have also the instruments to compute the similitude among objects in several ways. We have to do nothing more than using a *similarity measure*[11] available within the vector space, like the *cosine similarity*

$$\text{cosim}(\vec{x}_1, \vec{x}_2) = \frac{\vec{x}_1 \cdot \vec{x}_2}{\mid \vec{x}_1 \mid\mid \vec{x}_2 \mid} \tag{1.3}$$

where the numerator represents the *inner product* (also know as the *dot product*) of the vectors $\vec{x}_1$ and $\vec{x}_2$, while the denominator is the product of their lenghts. The denominator acts as a normalization factor, scaling each vector to the related unit vector. Thus, Equation 1.3 can be viewed as the inner product of the normalized versions of the two object vectors.
Obviously other distortion functions can be used, depending on the specific problem. In the sequel we will encounter some of these measures.

1.4 Conclusion

In this chapter we briefly introduced the core argument of this thesis, the *Clustering*. We saw that there are a lot of application domains interested in clustering, from business to science. Next, we summarized and schematized the clustering process. Finally we introduced the notations assumed in the rest of this thesis. We will take the clustering discussion again in the chapter 3 and the chapter 4 and will explain the problem more in detail in order to provide an exhaustive overview of the clustering techniques and highlighting the differences of each. However, we need first to provide the essentials on theoretical background and machine learning in the chapter 2.

[11] Also called *divergences*, *distortion functions*, *loss functions*.

CHAPTER

TWO

Machine learning essentials

«[...] La théorie des probabilités n'est, au fond, que le bon sens réduit au calcul.[a]*»*

[a] [...] The theory of probabilities is at bottom nothing but common sense reduced to calculus.
PIERRE-SIMON LAPLACE
(LAPLACE, 1812)

LEARNING, like intelligence, covers such a broad range of processes that is difficult to define properly. A dictionary definition include phrases such as "the acquisition of knowledge or skills through experience, practice, or study, or by being taught" or "modification of a behavioral tendency by experience". There are several parallels between animal or human learning and *machine learning*. In fact many machine learning techniques derive from the efforts of scientists like zoologists and biologists to make more precise their theories through computational models (Nilsson, 2005).[1]
Machine learning usually refers to the changes in systems that perform tasks associated with *Artificial Intelligence (AI)*, in such a manner that is expected future performance improves.
As already stated in the previous chapter, this thesis focus on a specific sub-area of machine learning, the *Clustering*. We will detailedly take clustering discussion again in the sequel. Meanwhile, in this chapter we provide the essentials on theoretical background and machine learning for the rest of this thesis.

[1] It seems likely also that the concepts and techniques being explored by machine learning researchers may illuminate certain aspects of biological learning.

2.1 Introduction

As a broad subfield of AI, machine learning is concerned with the design and development of algorithms and techniques that allow computers to *learn*. The major focus of machine learning research is to extract information from data automatically, by a variety of methods, such as computational methods, statistical methods, brain models and many others.
Machine learning has a wide spectrum of applications including natural language processing, syntactic pattern recognition, search engines, medical diagnosis, bioinformatics and cheminformatics, detecting credit card fraud, stock market analysis, classifying DNA sequences, speech and handwriting recognition, object recognition in computer vision, game playing and robot locomotion. Some machine learning systems attempt to eliminate the need for human intuition in the analysis of the data, while others adopt a collaborative approach between human and machine. However, human intuition *cannot* be entirely eliminated since the designer of the system must specify how the data are to be represented and what mechanisms will be used to search for a characterization of the data; so, machine learning can be viewed as an attempt to automate parts of the scientific method.

2.1.1 Machine learning and data mining

Machine learning is closely related to data mining, since the instruments used to tackle data mining problems have their solid fundamentals in the theoretical background of the machine learning research.
We can define *data mining* as the nontrivial extraction of implicit, previously unknown, and potentially useful information from data (Frawley et al., 1992). Data mining involves sorting through large amounts of data and picking out relevant information. It is usually employed in business intelligence applications, financial analysis, but it is increasingly used in the sciences (astrophysics, bioinformatics, etc.) to extract informations from the enormous data sets generated by modern experimental and observational methods.
The term data mining is often used to apply to the two separate processes of *knowledge discovery* and *prediction*. Knowledge discovery provides explicit information that has a readable form and can be understood by a user. Forecasting, or predictive modeling provides predictions of future events and may be transparent and readable in some approaches (e.g. rule based systems) and opaque in others such as neural networks. Moreover, some data mining systems such as neural networks are inherently geared towards prediction and pattern recognition, rather than knowledge discovery.

2.2 Taxonomy of machine learning research

Although machine learning systems can be classified according to different view points (Carbonell et al., 1984), a common choice is to classify such systems on the basis of the underlying learning strategies. In machine learning there are

two entities which play a crucial role: the *teacher* and the *learner*. The former is the entity which has the necessary knowledge to perform a given task; the latter indeed has to learn the knowledge to perform a task.
We can distinguish learning strategies by the amount of inference the learner performs on the information provided by the teacher. The greater the amount of inference that the learner is capable to perform, the lower the burden on the teacher. To build a taxonomy of machine learning we try to capture the notion of trade-off in the amount of effort required of the learner and of the teacher. Therefore, we indentify four different *learning types*: *rote learning*, *learning from instruction*, *learning by analogy* and *learning from examples*.
The *rote learning* consists in the direct implanting of new knowledge in the learner, such as learning by being programmed or learning by memorization of given facts. No inference is required on the part of the learner. *Learning from instructions* requires that the learner transforms the knowledge from the input language to an internal representation. The learner is required to perform some inference, but the most part of the cognitive burden remains with the teacher. *Learning by analogy* consists in acquiring new facts or skills by transforming and increasing existing knowledge. Such a system might be applied to convert an existing computer program into one that performs a closely-related function for which it was not originally designed.
The three types of learning just explained are not under the spotlight anymore because of the too low inference level of the learner and they have been ousted by the *learning from examples*. In this case the amount of inference performed by the learner is very high since it is able to generalize the learnt behavior on novel data. In fact, the *learning problem* can be described as finding a general rule that explains data given only a sample of limited size. Learning from examples has become so popular that it is simply called *learning* and in a similar way the examples are referred simply as *data*.[2]

2.2.1 Learning paradigms

Changing perspective, we can also organize learning[3] techniques based on the desired output. There are two main paradigms of learning: *Supervised Learning* and *Unsupervised Learning*. Other learning paradigms are *Reinforcement learning*, *Semi-supervised learning*, *Transduction* and so on, but we do not explore them.

2.2.1.1 Supervised learning

In *supervised learning* a learner is trained to build a function that can predict the correct output given the input, minimizing the errors as much as possible. The *teacher* (a training algorithm) use the *training set* to train the *learner*. The objects in the training set are already associated with the correct target values (outputs), e.g. *classes* or *real values*. After the training phase, the built function can be used to predict the output values for new ("untargeted") input patterns. The sets of these

[2] In the rest of the dissertation these conventions will be adopted.
[3] We recall that from now on the "learning" term refers to "learning from examples".

unlabeled inputs are called *test sets*. According to the type of the outputs, supervised learning can be distinguished in *Classification* and *Regression*. In Classification the output of the function resulting from the training stage is the predicted class label (usually an integer value) for the input object. This problem characterizes most pattern recognition tasks. A typical classification problem is to assign to a character bitmap the correct letter of the alphabet, i.e. the *Optical Character Recognition (OCR)*. In Regression the output of the function is a continuos value. One typical example of regression is the estimation of physical measurements values.

2.2.1.2 Unsupervised learning

If the data are only a sample of objects without associated target values, the problem is known as *unsupervised learning*. In unsupervised learning there is no teacher, it is the distribution and the nature of data that will determine the learner behavior and the related results. Hence a concise description of the data could be a set of groups or a probability density stating how likely it is to observe a certain object in the future. In unsupervised learning we are given a training sample of objects with the aim of extracting some structure from them. For instance, identifying indoor or outdoor images or extracting face pixels in an image. If some structure exists in training data, an unsupervised algorithm can take advantage of the redundancy and find a short description of data. A general way to represent data is to specify a similarity between any pairs of objects. If two objects share much structure, it should be possible to reproduce the data from the same prototype. This idea underlies *clustering algorithms* that form a rich subclass of unsupervised learning algorithms.
In addition to clustering algorithms, in unsupervised learning techniques there are algorithms whose aim is to represent high-dimensionality data in low dimension manifolds, trying to preserve the original information of data, such as *Principal Component Analysis (PCA)* (Bishop, 2006, sec. 12.1). These techniques are known under the name of *Multidimensional Scaling (MDS)* and are very useful to save storage space and/or face the *curse of dimensionality* (Bishop, 2006, sec. 1.4). Furthermore, we also have another[4] remarkable unsupervised technique, the *novelty detection*. Applications of this technique are fraud detection, process fault detection, homeland security and many other areas.

2.3 Frameworks for learning machines

In the following sections we aim at reviewing the major contributions which allowed the researchers over the past years to create robust frameworks for learning.
For the sake of brevity and consistency we only take into account the *statistical machine learning* and its contribution for designing of several learning frameworks.

[4]Other unsupervised techniques exist, but for the sake of brevity we do not list them here.

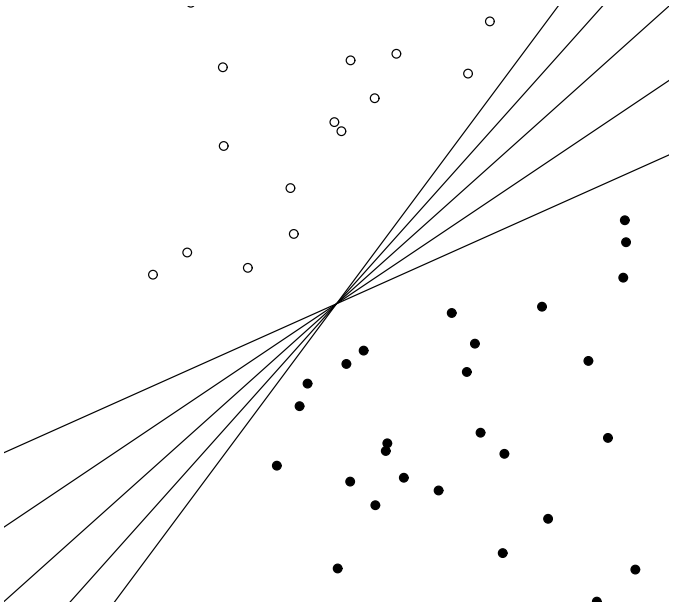


Figure 2.1: *A 2D classification problem. We can see that there is an infinite number of decision surfaces that separate two linearly separable classes. (Manning et al., 2007, fig. 14.8).*

So, we first provide (for historical motivations) a very brief overview on *Artificial Neural Networks (ANNs)* that thanks to statistics came back into the limelight in the 1986 (Rumelhart et al., 1986). Next, we present a general background on essential statistical instruments we will use in the subsequent chapters when we introduce the *MBI principle for Co-clustering* (Banerjee et al., 2007). In order to provide the necessary basics for introducing the novel *SVC* (Ben-Hur et al., 2001), we finally review detailedly the *SVMs* (Vapnik, 1995, 2000), considered the state-of-art of statistical machine learning research.

2.3.1 Linear and nonlinear separability

Learning machines became very interesting when solutions for nonlinear separable problems was found. In fact, real world problems in the majority of cases are nonlinear separable.
To understand better what it implies, let us generically define what *linear separable* means. For simplicity, we consider the case of a binary classification problem.

Definition 2.1 (Linear separability) *Given a binary classification problem in an n-dimensional space, the problem is said to be linearly separable if the two classes are separated by at least one single decision surface (or decision hyperplane).*

As showed in Figure 2.1, if linear separability holds, then there is an infinite number of linear separators. Once we have defined the linear separability, it is not hard to understand what we mean with *nonlinear separable* problem: such a problem (see Figure 2.2) cannot be separated by a single decision hyperplane and more

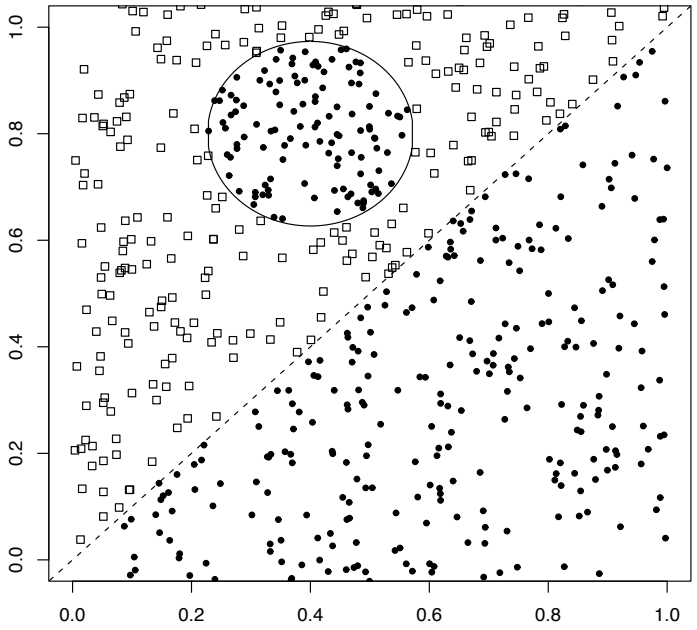


Figure 2.2: *A non linear problem. We can see that the linear separator (the dashed line) cannot put together all the objects represented by a fulfilled dot. (Manning et al., 2007, fig. 14.10).*

sophisticated machines are needed. Anyway, the currently available non-linear machines are not able to solve any kind of non-linear separable problem.

2.4 Artificial Neural Networks

The Artificial Neural Networks (ANNs) essentially born to emulate the extraordinary capabilities of the human brain and play a crucial role in the success of learning techniques. The first work on the ANNs was presented by McCulloch and Pitts (1943). Rosenblatt (1963) proposed a biologically motivated neural network, the so-called *Perceptron*, which can solve a wide range of simple classification problems with linearly separable classes. Minsky and Papert (1969) showed that the perceptron could not solve elementary nonlinear problems such as XOR. This negative result stopped practically active research in the field of neural networks for the next ten years and more. Rumelhart et al. (1986) proposed a generalization of the perceptron, the *Multi Layer Perceptron (MLP)*, which was capable to solve a wide class of learning problems. The algorithm proposed for the MLP training was the *Back Propagation*, originally based on the *least squares* principle, which have statistical fundamentals. MLP can be used for supervised learning problems, such as the *classification* and the *regression*. More recently, neural networks models for unsupervised learning have been proposed, such as *Neural Gas (NG)* (Martinetz and Schulten, 1991), *Topology Representing Networks (TRN)* (Martinetz and Schulten, 1994) or *Self-Organizing Maps (SOMs)* (Kohonen, 1995). The latter is widely used for clustering problems.

Even though the success of the Artificial Neural Networks (ANNs) in the last decade has been partially obscured by Support Vector Machines (SVMs), there is still much research effort in neural networks field.

2.5 Statistical machine learning

The main goal of statistical learning theory is to provide a framework for studying the problem of inference, making predictions, making decisions or constructing models from a set of data. Statistics laid the foundations for several learning frameworks, both supervised and unsupervised. Probability theory, decision theory, radial basis function methods, model-based methods, information theory: all of these disciplines are children of the statistics and they feed the machine learning research nowadays as in the past.
Most famous learning algorithms are *statistical machine learning* algorithms, such as the *Naive Bayes* classifier (Manning et al., 2007, chap. 13) or the *Expectation-Maximization (EM)* clustering algorithm (Dempster et al., 1977). Several learning algorithms based on so-called *loss functions*[5] have also statistical fundamentals, such as *K-means* clustering (MacQueen, 1967) or *kNN* classification (Manning et al., 2007, chap. 14). An example of information theory involvement in machine learning is the *Information Bottleneck (IB)* framework for clustering (Slonim and Tishby, 2000) or the *entropy-like* version of K-means (Teboulle et al., 2005).

2.5.1 Background and general method

Suppose the pattern x is a random variable whose probability distribution for category 1 is different than it is for category 2.[6] Specifically, suppose we have two probability distributions (perhaps probability density functions), $p(x|1)$ and $p(x|2)$. Given a pattern, x, we want to use statistical techniques to determine its category, i.e. to determine from which distribution it was drawn. These techniques are based on the idea of minimizing the *expectation* (or expected value) of a quantity, such as an error function (Nilsson, 2005; Vapnik, 2000).[7] Let us define what an expectation is.

Definition 2.2 (Expectation) *Let $f(x)$ be a function and let $p(x)$ be a probability distribution. The expectation of $f(x)$ is the average value of the function under the probability distribution $p(x)$ and it will be denoted by $\mathbb{E}[f]$.*

[5] Also known as *distortion functions*, *distances*, etc.

[6] For simplicity we choose to use only two categories, but the discussion can easily be generalized to multi-categories problems.

[7] Vapnik (2000, sec. 1.4) stated that the goal of the learner is the Empirical Risk Minimization (ERM). The empirical risk can be expressed in terms of a statistical expectation

$$R_{\text{emp}}(\alpha) = \frac{1}{n}\sum_{i=1}^{n} Q(x_i, \alpha) = \sum_{i=1}^{n} \nu_i Q(x_i, \alpha) = \mathbb{E}_\nu[Q(X, \alpha)]$$

where $Q(\cdot)$ is a generic loss function, n is the size of training set, $X = (x_i)_{i=1,2,\cdots,n}$ is the training set, $\nu = \frac{1}{n}$ is the uniform distribution, and α is a parameter.

For a discrete distribution, the expectation of $f(x)$ is given by

$$\mathbb{E}[f] = \sum_x p(x)f(x) \tag{2.1}$$

so that the average is weighted by the relative probabilities of the different values of x. In the case of continuous variables, expectations are expressed in terms of an integration with respect to the corresponding probability density

$$\mathbb{E}[f] = \int_x p(x)f(x)\mathrm{d}x \tag{2.2}$$

In developing a decision method, it is necessary to know the relative seriousness of the two kinds of mistakes that might be made: we might decide that a pattern really in category 1 is in category 2 and vice versa. We describe this information by a so-called *loss function*, $\lambda(i|j)$ for $i, j = 1, 2$. The function $\lambda(i|j)$ represents the loss incurred when we decide a pattern is in category i when really it is in category j. We assume here that $\lambda(1|1)$ and $\lambda(2|2)$ are both 0. For any given pattern, x, we want to decide its category in such a way that minimizes the expectation of this loss.
Given a pattern, x, if we decide category i, the expectation of the loss will be

$$\mathbb{E}_i[\lambda] = \lambda(i|1)p(1|x) + \lambda(i|2)p(2|x) \tag{2.3}$$

where $p(j|x)$ is the probability that given x, its category is j. Our decision rule will be to decide that x belongs to category 1 if $\mathbb{E}_1[\lambda] \leq \mathbb{E}_2[\lambda]$, and to decide category 2 otherwise.
Using the *Bayes' Rule* (Bishop, 2006, sec. 1.2.3) to get expressions for $p(j|x)$ in terms of $p(x|j)$ we arrive to the definition of the *maximum likelihood* which we present in chapter 4 in order to introduce the Expectation-Maximization (EM) properly.

2.6 Support Vector Machines

Support Vector Machines (SVMs) (Vapnik, 1995, 2000) became popular some years ago for solving supervised learning problems such as classification and regression. An important property of SVMs is that the indentification of the model parameters corresponds to a convex optimization problem, and so any local solution is also a global optimum. A SVM is a decision machine and so does not provide posterior probabilities[8] as some models of ANNs do (Bishop, 2006, sec. 5.7).
The original formulation of SVMs was made as a linear classifier.[9] Later these learning machines was extended to the nonlinear case, by means of the so-called *kernel trick*.

[8]There exist some revisited SVM models that provide posterior probabilities outputs, such as *Relevance Vector Machine (RVM)* (Bishop, 2006, sec. 7.2) or *Bayesian Least Squares Support Vector Machines (LS-SVM)* (Suykens et al., 2002, chap. 4).

[9]More precisely, a binary classifier. However, the extension to the multi-class case is straightforward.

2.6.1 Kernels essentials

The kernel concept was introduced into the field of pattern recognition by Aizerman et al. (1964) but was neglected for many years. With the coming of the SVMs the kernels was reintroduced in the machine learning. Since then, the use of the kernel trick (also known as *kernel substitution*) has been applied also in other algorithms for machine learning.
Let us define some basic facts in order to introduce kernels properly. [10]

Definition 2.3 (Positive semidefinite matrix) *A $n \times n$ symmetric[11] matrix $A = (a_{ij})_{i=1..n,j=1..n}$, $a_{ij} \in \mathbb{R}$ is called a* **positive semidefinite matrix** *if and only if*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j a_{ij} \geq 0 \tag{2.4}$$

for all $n \in \mathbb{N}$ and for all $c_1, c_2, \cdots, c_n \in \mathbb{R}$.

Theorem 2.1 *A matrix is positive semidefinite if and only if it is symmetric and has all eigenvalues greater than or equal to zero.*

Theorem 2.2 *Let $A = (a_{ij})$ be a symmetric $n \times n$ matrix. Then A is positive semidefinite if and only if*

$$(det(a_{ij})_{i,j \leq p}) \geq 0$$

where $p = 1, 2, \cdots, n$, i.e. all its minors have non-negatives determinants.

The above theorems underline the basic properties of a positive semidefinite matrix. Now we introduce the concept of *positive semidefinite kernel*, also known as *Mercer kernel*.[12]

Definition 2.4 (Mercer kernel) *Let $\mathcal{X} \subseteq \mathbb{R}$ be a non-empy set such that $\mathcal{X} = \{\vec{x}_1, \vec{x}_2, , \cdots, \vec{x}_n\}$. A mapping $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a* **positive semidefinite kernel** *or a* **Mercer kernel** *if and only if*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \phi(\vec{x}_i, \vec{x}_j) \geq 0$$

for all $n \in \mathbb{N}$, $\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n \subseteq \mathcal{X}$ and $c_1, c_2, \cdots, c_n \subseteq \mathbb{R}$.

The following results underline the basic properties of a Mercer kernel.

Theorem 2.3 *Let $\mathcal{X} \subseteq \mathbb{R}$ be a non-empy set such that $\mathcal{X} = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n\}$. A kernel $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive semidefinite if and only if it is symmetric and for every finite subset $\mathcal{X}_0 \subseteq \mathcal{X}$ the restriction ϕ_0 of ϕ such that $\phi_0 : \mathcal{X}_0 \times \mathcal{X}_0 \to \mathbb{R}$ is positive semidefinite.*

[10] The proofs of all theorems are omitted. More details in Cristianini and Shawe-Taylor (2000).

[11] In the most general case the definition is given for a Hermitian matrix. In the case of real numbers, a matrix is Hermitian if and only if it is symmetric.

[12] Negative kernels also exist but they are not used in machine learning so we do not review them here.

Theorem 2.4 *Let $\mathcal{X} \subseteq \mathbb{R}$ be a non-empy set such that $\mathcal{X} = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n\}$. Let $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive semidefinite kernel. The following condition holds*

$$\forall \vec{x} \in \mathcal{X}, \phi(\vec{x}, \vec{x}) \geq 0$$

Theorem 2.5 *Let $\mathcal{X} \subseteq \mathbb{R}$ be a non-empy set such that $\mathcal{X} = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n\}$. For any positive definite kernel $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ the following inequality holds*

$$\mid \phi(\vec{x_i}, \vec{x_j}) \mid^2 \leq \phi(\vec{x}_i, \vec{x}_i)\phi(\vec{x}_j, \vec{x}_j) \tag{2.5}$$

where $i, j = 1, 2, \cdots, n$ and $i \neq j$.

Now we introduce the result that allows to use Mercer kernels to make inner products.

Theorem 2.6 *Let K be a symmetric function such that for all $\vec{x}_i, \vec{x}_j \in \mathcal{X} = \{\vec{x}_1, \vec{x}_2, \cdots, , \vec{x}_n\} \subseteq \mathbb{R}$*

$$K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j) \tag{2.6}$$

where $\Phi : \mathcal{X} \to \mathcal{F}$ and $\mathcal{F}$ is an inner product higher dimensional space, called Feature Space*. The function K can be represented in terms of Equation 2.6 if and only if the* Gram *matrix $G = (K(\vec{x}_i, \vec{x}_j))_{i,j=1,2,\cdots,n}$ is positive semidefinite, i.e. $K(\cdot)$ is a Mercer kernel.*

The kernel function $K(\cdot)$ defines an *explicit* mapping if ϕ is known, otherwise the mapping is said to be *implicit*. In the majority of cases, the function ϕ is unknown, so we can implicitly perform an inner product in the feature space $\mathcal{F}$ by means of a kernel K.
Using nonlinear kernel transformations, we have a chance to transform a non separable problem in data space to a separable one in feature space (see Figure 2.3).

2.6.1.1 Valid Mercer kernels in $\mathbb{R}^n$ subspaces

There are several functions which are known to satisfy Mercer's condition for $\mathcal{X} \subseteq \mathbb{R}^n$. Some of them are

- *Linear* kernel: $K(\vec{x}, \vec{y}) = \vec{x}\vec{y}$
- *Polynomial* kernel: $K(\vec{x}, \vec{y}) = (\vec{x}\vec{y} + r)^k$, $r \geq 0$, $k \in \mathbb{N}$
- *Gaussian* kernel: $K(\vec{x}, \vec{y}) = \mathrm{e}^{-q\|\vec{x}-\vec{y}\|^2}$, $q > 0$
- *Exponential* kernel: $K(\vec{x}, \vec{y}) = \mathrm{e}^{-q\|\vec{x}-\vec{y}\|}$, $q > 0$
- *Laplacian* kernel: $K(\vec{x}, \vec{y}) = \mathrm{e}^{-q|\vec{x}-\vec{y}|}$, $q > 0$

where $\|\cdot\|$ is the L2 metric (also known as *Euclidean distance*) and $|\cdot|$ is the L1 metric (also known as *Manhattan distance*).
In addition, we can build other kernel functions: given two Mercer kernels K_1 and K_2, we can construct new Mercer kernels by properly combining K_1 and K_2 (Cristianini and Shawe-Taylor, 2000, sec. 3.3.2).

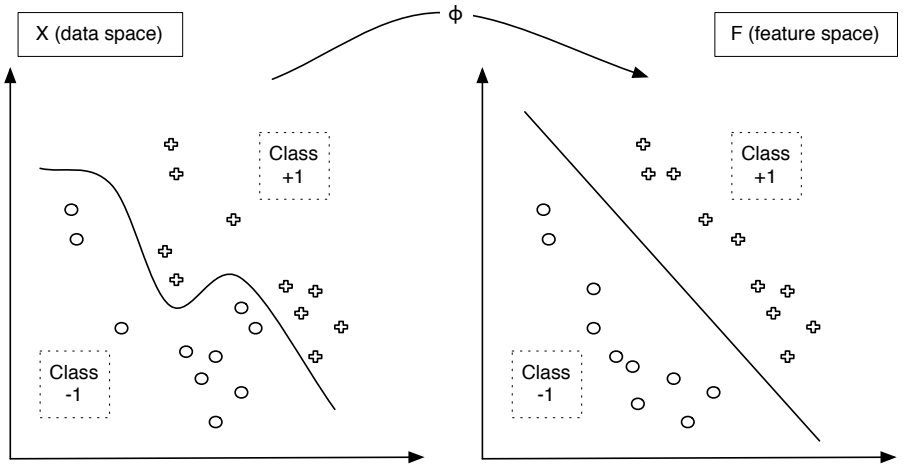


Figure 2.3: *A nonlinear separable problem in the data space $\mathcal{X}$ that becomes linear separable in the feature space $\mathcal{F}$.*

In polynomial kernels, the parameter k is the *degree*. In the exponential-based kernels (and others), the parameter q is called *kernel width*. The kernel width has different mathematical meaning depending on the kernel: in the case of the Gaussian kernel, it is a function of the *variance*

$$q = \frac{1}{2\sigma^2}$$

Kernel width[13] is a general term to indicate the scale at which data is probed. Because of the aforementioned mathematical differences, a proper kernel width for a particular kernel function can be not proper also for another kernel in context of the same problem.

2.6.2 Maximal margin classifier

The "margin" concept is a first important step towards understanding the formulation of Support Vector Machines (SVMs). As we can see in Figure 2.1, there exist several separating hyperplanes that separate the data in two classes and some classifiers (such as the perceptron) just find one of them as solution. On the other hand, we can also define a unique separating hyperplane according to some criterion. The SVMs in particular define the criterion to be looking for a decision surface that is maximally far away from any data points. This distance from the decision surface to the closest data point determines the *margin* of the classifier. An SVM is designed to maximize the margin around the separating hyperplane. This necessarily means that the decision function for an SVM is fully specified by a usually small subset of the data which defines the position of the separator. These points are referred to as the *support vectors*. The data points other than support vectors play no part in determining the decision surface.

[13] As we will see in the sequel, the kernel width is one of the SVM training hyper-parameters.

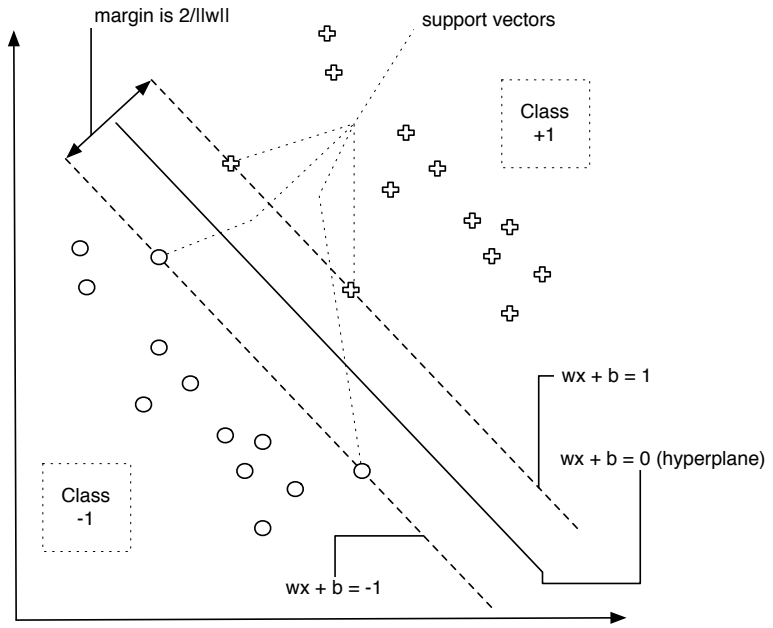


Figure 2.4: *Linear classification: definition of a unique hyperplane (the solid line) illustrated in a 2-dimensional data space. The (geometric) margin is the distance between the dashed lines.*

A decision hyperplane can be defined by normal vector $\vec{w}$, therefore $\vec{w}\vec{x} = 0$ for all points $\vec{x}$ on the hyperplane. To choose among all the hyperplanes that are normal to the vector $\vec{w}$, we also specify an intercept term b. So, a hyperplane is represented by the following equation

$$\vec{w}\vec{x} = b \tag{2.7}$$

Now we can distinguish two types of margin.

Definition 2.5 (Functional margin) *The* **functional margin** γ_{f_i} *of an example* $\vec{x}_i \in \mathcal{X} = \{\vec{x}_k, y_k\}_{k=1,2,\cdots,n}$ *with respect to a hyperplane* $\vec{w}\vec{x} = b$ *is the product* $y_i(\vec{w}\vec{x}_i + b)$.

Definition 2.6 (Geometric margin) *The* **geometric margin** γ_{g_i} *of an example* $\vec{x}_i \in \mathcal{X} = \{\vec{x}_k, y_k\}_{k=1,2,\cdots,n}$ *with respect to a hyperplane* $\vec{w}\vec{x} = b$ *is the product* $y_i(\frac{\vec{w}}{\|\vec{w}\|}\vec{x}_i + \frac{b}{\|\vec{w}\|})$.

The problem of the functional margin is that we can make it as big as we wish by simply scaling up $\vec{w}$ and b by a multiplication factor. On the contrary, the geometric margin is invariant to any scaling factors, because it is always normalized to the unit vector, such that $\|\vec{w}\vec{x}_k + b\| = 1$. In this case the geometric margin $\gamma_g = 2/\|w\|$. Therefore, from now on we will work with geometric margin and it will be simply referred as "margin" γ.

2.6.3 Linear Support Vector Machines

Consider a training set $\{\vec{x}_k, y_k\}_{k=1,2,\cdots,n}$ with input patterns $\vec{x}_k \in \mathbb{R}^d$ and output data $y_k \in \{-1, +1\}$ and the linear classifier[14]

$$y(\vec{x}) = \text{sgn}(\vec{w}\vec{x} + b) \tag{2.8}$$

When the data of two classes are separable we can say

$$\begin{cases} \vec{w}\vec{x}_k + b \geq +1, & \text{if } y_k = +1 \\ \vec{w}\vec{x}_k + b \leq -1, & \text{if } y_k = -1 \end{cases} \tag{2.9}$$

These two sets of inequalities can be combined into a single set as follows

$$y_k[\vec{w}\vec{x}_k + b] \geq 1, \quad k = 1, 2, \cdots, n \tag{2.10}$$

Now we can formulate the SVM training problem. First we formulate the constrained optimization problem in the primal space and then we reformulate it in the Lagrangian dual form.[15]

Problem 2.1 *We want to maximize the margin subject to the fact that all training points need to be correctly classified.*

So, the *primal form* in w of the Problem 2.1 is

$$\max_{w,b} \frac{2}{\|\vec{w}\|} \tag{2.11}$$

subject to

$$y_k[\vec{w}\vec{x}_k + b] \geq 1, \quad k = 1, 2, \cdots, n$$

which is equivalent to

$$\min_{w,b} \frac{\|\vec{w}\|}{2} \tag{2.12}$$

subject to

$$y_k[\vec{w}\vec{x}_k + b] \geq 1, \quad k = 1, 2, \cdots, n$$

We know that $\|\vec{w}\| = \sqrt{\vec{w} \cdot \vec{w}}$, so we can rewrite Equation 2.12

$$\min_{w,b} \frac{1}{2}\vec{w}\vec{w} \tag{2.13}$$

subject to

$$y_k[\vec{w}\vec{x}_k + b] \geq 1, \quad k = 1, 2, \cdots, n$$

[14] The sign function $\text{sgn}(\cdot)$ is a logical function which extracts the sign of a real number.

[15] The *Lagrangian Strong Duality Theorem* assures that an optimal solution of the dual is also an optimal solution for the primal problem and vice versa (Boyd and Vandenberghe, 2004, chap. 5).

as minimizing $\|\vec{w}\|$ is the same to minimize $\|\vec{w}\|^2$.
However, most real-life problem are non-separable either in linear or nonlinear sense. This is due to the fact that the underlying data distribution overlap. In order to allow data separation in such situations, we should consider to tolerate some misclassifications. So, we can extend the SVM formulation to the non-separable case by taking additional slack variables $\{\xi_k\}_{k=1}^n$ in the problem formulation

$$\min_{w,b,\xi} \frac{1}{2}\vec{w}\vec{w} + C\sum_{k=1}^{n}\xi_k \tag{2.14}$$

subject to

$$y_k[\vec{w}\vec{x}_k + b] \geq 1, \quad k = 1, 2, \cdots, n$$
$$\xi_k \geq 0, \quad k = 1, 2, \cdots, n$$

where C is a positive real constant which acts as *regularization* term and provides a way to control overfitting: the lower the C the higher the number of points that are moved around to tune the margin. In fact, now the optimization problem trades off how fat it can make the margin versus how many points have to be moved around to allow this margin. By setting $\xi_k > 0$ we allow a margin less than 1 for the point x_k, but we pay a penalty of $C\xi_k$ for having done this.
The Lagrangian (Boyd and Vandenberghe, 2004, chap. 5) for Equation 2.14 is

$$\mathcal{L}(w, b, \xi; \beta, \mu) = \frac{1}{2}\vec{w}\vec{w} + C\sum_{k=1}^{n}\xi_k - \sum_{k=1}^{n}\beta_k(y_k[\vec{w}\vec{x}_k + b] - 1 + \xi_k) - \sum_{k=1}^{n}\mu_k\xi_k \tag{2.15}$$

with Lagrange multipliers $\beta_k \geq 0, \mu_k \geq 0$ for $k = 1, 2, \cdots, n$. The solution is characterized by the saddle point of the Lagrangian

$$\max_{\beta,\mu}\min_{w,b,\xi}\mathcal{L}(w, b, \xi; \beta, \mu). \tag{2.16}$$

We obtain

$$\begin{cases} \frac{\partial\mathcal{L}}{\partial w} = 0 \quad \rightarrow & w = \sum_{k=1}^{n}\beta_k y_k \vec{x}_k \\ \frac{\partial\mathcal{L}}{\partial b} = 0 \quad \rightarrow & \sum_{k=1}^{n}\beta_k y_k = 0 \\ \frac{\partial\mathcal{L}}{\partial \xi_k} = 0 \quad \rightarrow & 0 \leq \beta_k \leq C, \quad k = 1, 2, \cdots, n \end{cases} \tag{2.17}$$

with resulting classifer

$$y(\vec{x}) = \text{sgn}\left(\sum_{k=1}^{n}\beta_k y_k \vec{x}_k \vec{x} + b\right) \tag{2.18}$$

By substituting the expression for w (Equation 6.8) in the Equation 2.15 we obtain the Quadratic Programming (QP) problem as the Lagrangian *dual problem*

$$\max_{\beta} -\frac{1}{2}\sum_{k,l=1}^{n}\beta_k\beta_l y_k y_l \vec{x}_k\vec{x}_l + \sum_{k=1}^{n}\beta_k \tag{2.19}$$

subject to

$$\sum_{k=1}^{n}\beta_k y_k = 0$$
$$\forall k = 1, 2, \cdots, n \;\; 0 \leq \beta_k \leq C$$

The problem is not solved in w anymore, but in β. This QP problem has a number of interesting properties. First, we have a *global and unique solution*. Next, the solution vector $\vec{\beta}$ is *sparse*, i.e. only the beta values associated to support vectors are non-zero, so we can rewrite Equation 2.18 as follows

$$y(\vec{x}) = \text{sgn}\left(\sum_{k=1}^{N_{SV}}\beta_k y_k \vec{x}_k\vec{x} + b\right) \tag{2.20}$$

where N_{SV} is the number of support vectors. Finally, we note that the solution vector $\vec{\beta}$ grows with the number of data points n. On the contrary the size of w for the primal problem is fixed and does not depend on the number of data points. This means that for large data sets it might be advantageous to solve the primal problem, but the dual problem has the advantage of being *robust with respect to dimensionality*, i.e. the number d of features.

2.6.4 Nonlinear Support Vector Machines

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be the data space where the data points lie. Let $\phi : \mathcal{X} \to \mathcal{F}$ a non-linear transformation from $\mathcal{X}$ to a higher-dimensional feature space[16] $\mathcal{F}$, where a non separable problem could become separable (see Figure 2.3). By means of this nonlinear mapping we can easily extend the Support Vector Machine (SVM) formulation to the nonlinear case, replacing data points with the corresponding transformations. So, the Equation 2.10 becomes

$$y_k[\vec{w}\phi(\vec{x}_k) + b] \geq 1, \quad k = 1, 2, \cdots, n \tag{2.21}$$

and the optimization problem (Equation 2.14) results in

$$\min_{w,b,\xi} \frac{1}{2}\vec{w}\vec{w} + C\sum_{k=1}^{n}\xi_k \tag{2.22}$$

subject to

$$y_k[\vec{w}\phi(\vec{x}_k) + b] \geq 1, \;\; k = 1, 2, \cdots, n$$
$$\xi_k \geq 0, \;\; k = 1, 2, \cdots, n$$

[16] From a neural network perspective, the feature space corresponds to one or more high dimensional *hidden layers*. In fact, in pattern recognition the term *feature space* is often used with another meaning of input space. Nevertheless we will use the term "feature space" in the sequel because it is the "official" term in machine learning.

The Lagrangian dual problem (Equation 2.19) becomes

$$\max_{\beta} -\frac{1}{2}\sum_{k,l=1}^{n}\beta_k\beta_l y_k y_l \phi(\vec{x}_k)\phi(\vec{x}_l) + \sum_{k=1}^{n}\beta_k \tag{2.23}$$

$$\text{subject to}$$

$$\sum_{k=1}^{n}\beta_k y_k = 0$$

$$\forall k = 1, 2, \cdots, n \;\; 0 \leq \beta_k \leq C$$

By means of a Mercer kernel we can rewrite the previous Quadratic Programming (QP) as

$$\max_{\beta} -\frac{1}{2}\sum_{k,l=1}^{n}\beta_k\beta_l y_k y_l K(\vec{x}_k, \vec{x}_l) + \sum_{k=1}^{n}\beta_k \tag{2.24}$$

$$\text{subject to}$$

$$\sum_{k=1}^{n}\beta_k y_k = 0$$

$$\forall k = 1, 2, \cdots, n \;\; 0 \leq \beta_k \leq C$$

where $K(\vec{x}_k, \vec{x}_l) = \phi(\vec{x}_k) \cdot \phi(\vec{x}_l)$. Consequently, the nonlinear SVM classifier takes the form

$$y(\vec{x}) = \text{sgn}\left(\sum_{k=1}^{n}\beta_k y_k K(\vec{x}_k, \vec{x}) + b\right) \tag{2.25}$$

The *kernel trick* allow us to implicitly work in a higher-dimensional feature space without explicitly knowing the mapping ϕ.
The previously mentioned advantages of the dual form of the problem still hold.

2.7 Conclusion

In this chapter we introduced the machine learning theory and its most important instruments. We saw a very brief historical overview on Artificial Neural Networks (ANNs), basic concepts of the statistical learning theory and the state of art of the kernel methods for learning: the nonlinear Support Vector Machines (SVMs). As previously said, the rest of this thesis is based on the *Clustering*, a specific sub-area of the unsupervised branch of the machine learning techniques. The concepts presented in this chapter are crucial for a full understanding of the two state-of-the-art techniques for data clustering we present in chapter 5 and chapter 6.
Meanwhile, in the chapter 3 we will analyze in depth the clustering and the main problems we have to face in developing and applying clustering techniques.

CHAPTER

THREE

Clustering and related issues

«The best causes tend to attract to their support the worst arguments.»

RONALD AYLMER FISHER
(FISHER, 1956, P. 31)

CLUSTERING was introduced in chapter 1 as the core argument of this dissertation. Within this chapter we will go deeper into the matter, providing a formal problem statement and a taxonomy on clustering techniques. Furthermore we will treat also the main clustering problems. Clustering has many issues that have to be addressed to make any clustering solution successful. Some of them are so ubiquitous that they are not specific issues of clustering, but they involve the whole machine learning and data mining panorama. Other ones are resolved by specific clustering techniques which are often designed with particular issues in mind. Common problems are the assessment of the results, the estimation of the right number of clusters, the high dimensionality of data, the noise, the sparseness of the data matrix, the outliers handling and the capability of treating clusters of arbitrary shape. Side by side, we review also the most common solutions for each issue.

3.1 Problem statement

Like all machine learning techniques, the clustering problem can be formalized as an optimization problem, i.e. the minimization or maximization of a function subject to a set of constraints. Starting from the overview of the clustering process (see section 1.2) we can generally define the goal of a clustering algorithm as follows.

Given

(i) a dataset $\mathcal{X} = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n\}$

(ii) the desired number of clusters k

(iii) a function f that evaluates the quality of clustering

we want to compute a mapping

$$\gamma : \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, k\} \tag{3.1}$$

that minimizes[1] the function f subject to some constraints.
The function f that evaluates the clustering quality are often defined in terms of *similarity*[2] between objects and it is also called *distortion function* or *divergence*. The similarity measure is the key input to a clustering algorithm.
Hence we build an objective function by means of divergence function and some constraints (including the number of aimed clusters) and our goal is to minimize[3] it finding the suitable γ application. In most cases, we also demand that the mapping γ is surjective, i.e. no cluster have to be empty. In this case we can formally state that the goal of a clustering algorithm is to build a partition of the initial dataset $\mathcal{X}$ which have cardinality k.
In the next section we will see that this is only one of the possible formal definition for the clustering problem, though it is the most common and widespread one (the *flat partitional clustering*).

3.2 Taxonomy on clustering techniques

Categorization of the clustering algorithms is neither straightforward nor canonical. In reality, whatever classification we do, the various classes of algorithms often overlap, so we provide only the most important categorizations for clustering algorithms.

3.2.1 Flat clustering and hierarchical clustering

An important classification divides the clustering algorithms in two main classes: *Flat Clustering* and *Hierarchical Clustering* (Manning et al., 2007, chap. 16). The flat clustering creates a flat set of clusters without any explicit structure or information that would relate clusters to each other (see Figure 3.1). One crucial point in a flat clustering algorithms is the choice of the number k of clusters to provide as input parameter, i.e. the *cardinality* of clustering: often k is nothing more than a good guess based on experience or domain knowledge[4]. On the contrary, hierarchical clustering builds a cluster hierarchy or, in other words, a tree of clusters, also known as a *dendrogram*.

[1] Or in other cases maximizes.

[2] The similarity could be a proximity measure, a probability, etc.

[3] We know that every maximization problem can be translated in an equivalent minimization problem.

[4] However some heuristic methods have been developed to estimate a good value for k.

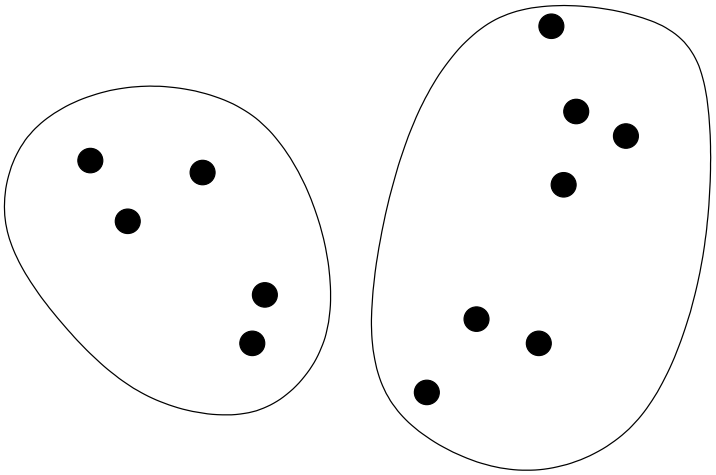


Figure 3.1: *A classic flat clustering results.*

Every cluster node contains child clusters; sibling clusters partition the points covered by their common parent (see Figure 3.2). Such an approach allows exploring data on different levels of granularity. Hierarchical clustering methods are further categorized in *agglomerative* and *divisive*. The former is a bottom-up approach that starts with singleton[5] clusters and recursively merges two or more most appropriate clusters. The latter is a top-down approach that starts with one cluster containing all data points and then recursively splits the most appropriate cluster. In both cases the process goes on until a user-defined stopping criterion[6] is achieved. The main advantage of the hierarchical clustering over the flat clustering is that we can avoid to specify the number of clusters as input of the algorithm. In fact, we can build a different stopping criterion, even though the specification of such criterion in this case is the main drawback of the hierarchical approach. Another advantage of the hierarchical clustering is the possibility of examining the clustering results at different levels of granularity, which results in more flexibility. A disadvantage could be the necessity of extending similarity measures to handle similarity among clusters in order to have a way to choose what cluster is appropriate to merge or to split.
The classical approaches to hierarchical clustering use *Linkage Metrics* in order to deal with clusters similarity measures. Hierarchical clustering based on the linkage metrics results in clusters of convex shapes, but some effort was spent to develop hierarchical clustering algorithms able to treat cluster of arbitrary shapes (Guha et al., 1998; Karypis et al., 1999).
In hierarchical clustering our regular point-by-attribute data representation is sometimes of secondary importance. Instead, hierarchical clustering frequently deals with the $n \times n$ matrix of similarity measures between training points; it is sometimes called *connectivity* matrix and linkage metrics are constructed from

[5] A cluster containing only one point.
[6] Frequently, a bound on requested number k of clusters.

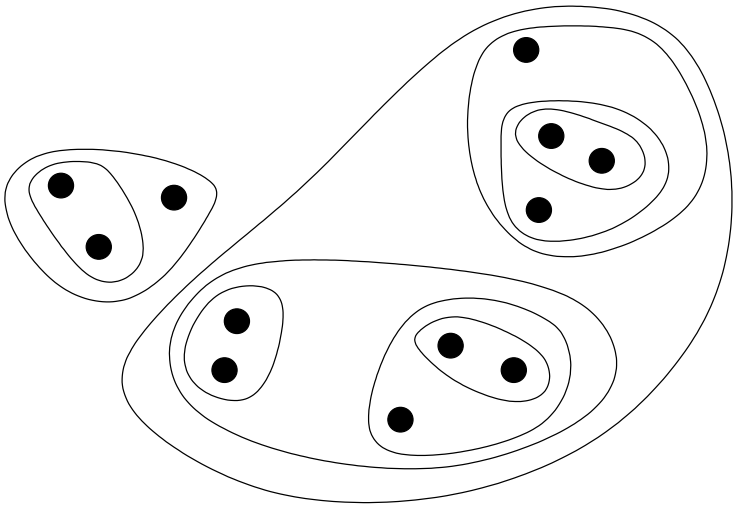


Figure 3.2: *An illustration of typical hierarchical clustering results.*

elements of such a matrix.

3.2.2 Partitional and non-partitional clustering

Another important subdivision of the clustering algorithms is between *Partitional Clustering* and *Non-partitional Clustering* (Manning et al., 2007, chap. 16). The algorithms which belong to the first class perform a *hard assignment*: each object become a member of exactly one cluster. In this case the assumption of the cluster intersection emptiness (see Equation 1.2) holds and therefore we often call this class of algorithms *Hard Clustering* too. Instead the algorithms belonging to the non-partitional clustering class perform a so called *Soft Clustering* (or *Fuzzy Clustering*) because they can assign points to more than one cluster using a membership function (Zadeh, 1965), i.e. they compute a soft assignment. In this case the Equation 1.2 does not hold anymore.

3.2.3 Flat partitional clustering

The hard clustering is widespread among all application fields alongside the flat clustering model, much more than other clustering model combinations. Therefore, one of most popular clustering schemes is the *Flat Partitional Clustering*, i.e. a clustering scheme where the final goal is to build a partition of a prefixed cardinality k of the input data set X.
Because checking all possible subset system is computationally unfeasible, certain greedy heuristics are used in the form of *iterative optimization*. Specifically, this involves a different relocation scheme that iteratively moves points from one cluster to the other. Unlike traditional hierarchical models, in which clusters are not revisited after being constructed, relocation algorithms gradually improve clusters. With appropriate data, this results in high quality clusters.

One approach to data partitioning is to take a conceptual point of view that identifies the cluster with a certain model whose unknown parameters have to be found. More specifically, *probabilistic* models assume that the data comes from a mixture of several populations whose distributions and priors we want to find.
Another approach starts with the definition of an objective function depending on a partition. Using unique cluster representatives makes the computation of the objective function linear in n. Depending on how the representatives are constructed, iterative partitioning algorithms are further subdivided in *k-medoids* and *k-means* methodologies. In *k*-medoids methods each cluster is represented by the most appropriate data point within the cluster itself; this results in a lower sensitiveness to the presence of outliers, because the choice of the medoids is dictated by the location of the predominant fraction of points inside the cluster. Another advantage of the *k*-medoids methods is the capability of working with attributes of any type. In *k*-means case, a cluster is represented by its *centroid*. A centroid of a cluster $\mathcal{C}$ is computed as the average[7] or center of mass of its members.[8]
This works well only with numerical attributes and the presence of just a single outlier can affect negatively the process. On the other hand, centroids have the advantage of clear geometric and statistical meaning.
The name of the *k*-means methods comes directly from the name of the first algorithm of this type, the *K-means* (MacQueen, 1967).

3.2.4 One-way clustering and multi-dimensional clustering

Most clustering algorithms focus on one-way clustering, i.e. the clustering along one dimension of the $n \times d$ input data matrix, based on the similarities along the second dimension. For example, text documents (along the rows of a data matrix) are usually grouped based upon their word distributions (along the columns of the same data matrix). Recently much effort in the clustering research was dedicated to the simultaneous clustering along multiple dimensions (Banerjee et al., 2004a; Cai et al., 2005; Cheng and Church, 2000; Cho et al., 2004a; Dhillon, 2001; Dhillon et al., 2003b; Guan et al., 2005; Long et al., 2005; Qiu, 2004; Tchagang and Tewfik, 2005; Yang et al., 2003), notwithstanding this clustering approach was already introduced in the early 70s by Hartigan (1972).
Simultaneous clustering are usually performed along two dimensions, exploiting the clear duality between rows and columns of a data matrix. Such a type of clustering has many advantages over the classic one-way clustering. First, a unique characteristic of this approach is the ability to reveal interplay among clusters along all dimensions. Furthermore, as we will show in next chapters, this clustering technique has the ability to perform an implicit dimensionality reduction and at the same time to perform well with sparse data matrix.

[7] Usually a weighted average.
[8] See section 4.1 for more details.

3.2.5 Other clustering types

There are many other classifications of the clusterings methodologies, such as *Valley finding clustering* versus *Peak finding clustering* or *Statistical clustering* versus *Conceptual clustering*. Such classifications are often not clearly outlined (therefore we may encounter significant overlapping) or are too complex. Therefore, for the sake of brevity and consistency, we choose to report only the most significant classifications from our viewpoint.

3.3 Evaluation of clustering

One of the most important issues in clusters analysis is the evaluation of the clustering results. Evaluating clustering results is the analysis of the output to understand how well it reproduces the original structure of the data. However, the evaluation of clustering results is the most difficult task within the whole clustering workflow. In most experimental evaluations two-dimensional data sets are used in order that the reader is able to visually verify the validity of the results (i.e. how well the clustering algorithm discovered the clusters of the data set). It is clear that visualization of the data set is a crucial verification of the clustering results, but in the case of large multidimensional datasets (e.g. more than three dimensions) effective visualization of the dataset would be difficult. Moreover the perception of clusters using available visualization tools is a difficult task for humans that are not accustomed to higher dimensional spaces.
We have to employ different techniques to perform the *cluster validation*. In general terms, there are three approaches to investigate the clustering quality (Halkidi et al., 2001; Zaïane et al., 2002). The first is based on *internal criteria*, i.e. the optimization criterion which the involved clustering algorithm is based on. Internal criteria are the only means we have in case we are going to tackle a completely new domain, where neither labels nor gold standards are available. The second one in based on *external criteria*; these imply clustering evaluation by means of pre-specificed structure, which is imposed on a dataset and reflects our intuition about clustering structure of such a dataset. An external criterion could be used also involving gold standards. Finally, the third approach is based on *relative criteria*. Here the basic idea is the evaluation of a clustering structure by comparing it to other clustering schemes resulting by the same algorithm but with different input parameter values.

3.3.1 Internal criteria

Typical objective functions in clustering formalize the goal of attaining high intra-cluster similarity (objects within a cluster are similar) and low inter-cluster similarity (objects from different clusters are dissimilar). This is an internal criterion for the quality of a clustering. Typical internal criteria are the minimization of the average squared distance of objects from their clusters centers[9](Hartigan and

[9]The centroid, the medoid, etc.

Wong, 1979), the minimization of the loss in mutual information between the input data matrix and one of its approximations based on the clustering (Dhillon et al., 2003b), the minimization of the normalized cut of a graph (Dhillon, 2001), the maximization of the Renyi's entropy (Gokcay and Principe, 2000).
But good scores on an internal criterion do not necessarily translate into good effectiveness in an application, so a valid alternative would be the direct evaluation in the application of interest. Unfortunately this is unfeasible in the majority of cases.

3.3.2 External criteria

User-based evaluation is nearly always too expensive. As a surrogate for user judgments, we can use a set of classes in evaluation benchmark or gold standard. The gold standard is ideally produced by human judges with a good level of inter-judge agreement (Cohen, 1960). We can then compute an external criterion that evaluates how well clustering matches the gold standard classes.
In this section we introduce several types of external criteria. First, we find *purity*, a simple and transparent evaluation measure; to compute purity each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned objects and dividing by n. Formally

$$\text{purity}(\Omega, \Gamma) = \frac{1}{n} \sum_k \max_j \mid \omega_j \cap c_j \mid \tag{3.2}$$

where $\Omega = \{\omega_1, \omega_2, \cdots, \omega_k\}$ is the set of clusters and $\Gamma = \{c_1, c_2, \cdots, c_J\}$ is the set of classes.
Bad clusterings have purity values close to 0, whereas a perfect clustering has a purity of 1.0; high purity is easy to achieve in case of large number of clusters, in particular purity is 1.0 if each object gets its own cluster. Thus, we cannot use this criterion to trade off the quality of the clustering against the number of clusters.
Another measure which allows us to make such a tradeoff is the *Normalized Mutual Information (NMI)*

$$\text{NMI}(\Omega, \Gamma) = \frac{\text{I}(\Omega; \Gamma)}{[\text{H}(\Omega) + \text{H}(\Gamma)]/2} \tag{3.3}$$

where I is the *mutual information*

$$\text{I}(\Omega; \Gamma) = \sum_k \sum_j P(\omega_k \cap c_j) \log \frac{P(\omega_k \cap c_j)}{P(\omega_k) P(c_j)} \tag{3.4}$$

and H is the *entropy*

$$\text{H}(\Omega) = -\sum_k P(\omega_k) \log P(\omega_k) \tag{3.5}$$

and where $P(\omega_k)$, $P(c_j)$ and $P(\omega_j \cap c_j)$ are the probabilities of an object being in ω_k, c_j and in intersection of ω_k and c_j respectively.

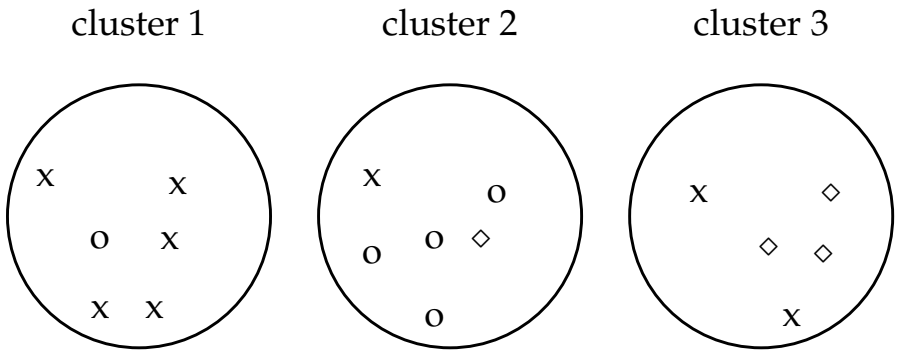


Figure 3.3: *Purity is an external criterion for evaluation of cluster quality. Majority class and number of members of the majority class for the three clusters are: X, 5 (cluster 1); circle, 4 (cluster 2); diamond, 3 (cluster 3). Purity is* $(1/17) * (5 + 4 + 3) \approx 0.71$ *(Manning et al., 2007, fig. 16.4).*

The Equation 3.4 measures the amount of information by which our knowledge about the classes increases when we are told what clusters are. This term reaches a minimum of 0 if the clustering is random with respect to the classes and so the Equation 3.3. In that case, knowing that an object is in a particular cluster does not give us any new information about what class it might be in. Maximal mutual information is reached for a perfect clustering, i.e. a clustering that exactly recreates the classes, but also if clusters of this perfect clustering are further split into smaller clusters. In particular, a clustering with $K = N$ singleton clusters has the maximum mutual information, so the mutual information alone has the same problem which affects the purity measure: it does not penalize large clustering cardinalities and thus does not formalize our bias that fewer clusters are better. This is why in the Equation 3.3 we add a normalization term at the denominator to fix the problem, since the entropy tends to increase with the number of clusters. For istance, $\mathrm{H}(\Omega)$ reaches its maximum for $K = N$ which ensures that the NMI is low for $K = N$. Because NMI is normalized, we can use it to compare clusterings with different number of clusters.

3.3.2.1 Criteria derived from classification tasks

An alternative to the information theoretic interpretation of clustering previously stated is to view it as a series of decisions, one for each of the $N(N-1)/2$ pairs of objects in the collection. We want to assign two objects to the same cluster if and only if they are similar. A *True Positive (TP)* decision assigns two similar objects to the same cluster, whereas a *True Negative (TN)* decision assigns two dissimilar objects to different clusters, so there are two errors we can commit: first, a *False Positive (FP)* decision, which assigns two dissimilar objects to the same cluster; second, a *False Negative (FN)*) decision, which assigns two similar objects to different clusters. The *Rand Index (RI)* is the external criterion which measures the percentage of correct decisions, i.e. it is simply the *accuracy* value

used in the evaluation of the classification tasks

$$\mathrm{RI} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{3.6}$$

The main drawback of the RI is that it gives equal weight to false positives and false negatives. In some situations, separating similar objects is worse than putting dissimilar objects in the same cluster. To tackle these situations, we can use the *F measure*, also used in the evaluation of the classification tasks, which is the harmonic mean of *P* and *R*.
The precision is the percentage of positive predictions that are correct

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3.7}$$

whereas the recall is the percentage of positive labeled instances that were predicted as positive

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.8}$$

So, the F-measure is

$$\mathrm{F}_\beta = \frac{1}{\alpha\frac{1}{\mathrm{P}} + (1-\alpha)\frac{1}{\mathrm{R}}} = \frac{(\beta^2+1)\mathrm{PR}}{\beta^2\mathrm{P} + \mathrm{R}} \tag{3.9}$$

where $\beta^2 = 1 - \alpha/\alpha$. Choosing a $\beta > 1$ we can penalize false negatives more strongly than false positives, i.e. we can give more weight to recall.

In some application domains could be useful some variations of the Precision-Recall measures. An example is the couple *Sensitivity-Specificity*. Sensitivity is just another name for the recall, while the specificity is a dual form of the recall

$$\text{Specificity} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{3.10}$$

that is, the specificity is the percentage of negative labeled instances that were predicted as negative.
There is also another name for the recall, usually used in the astrophysics data mining: *completeness*. The completeness is often used together with the *contamination* measure (Donalek, 2006)

$$\text{Contamination} = \frac{\mathrm{FP}}{\mathrm{TP} + \mathrm{FP}} \tag{3.11}$$

In other words, the contamination is the percentage of the objects incorrectly put in a given class.
Note that the contamination is the complement of the precision

$$1 - \mathrm{P} = 1 - \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{\mathrm{FP}}{\mathrm{TP} + \mathrm{FP}} = \text{Contamination} \tag{3.12}$$

3.3.3 Relative criteria

With relative criteria the challenge is to characterize the clustering result in a way that tells us the quality of the clustering. Naturally, there is a grey line between measures used by clustering algorithms to determine where to join or split clusters and indices proposed to determine if that was good. When we construct an index for evaluation of clustering there are two general criteria to take into account

- *Compactness*. The members of each cluster should be as close as possible to each other. A common measure of compactness is the variance, which should be minimized in order to achieve the best compactness.
- *Separation*. The clusters themselves should be widely spaced. There are three common approaches measuring the distance between clusters:
  - *Single linkage*. It measure the distance between the closest members of two clusters.
  - *Complete linkage*. It measure the distance between the most distant members of two clusters.
  - *Centroids comparison*. It measure the distance between the centers of two clusters.

A number of validity indices have been developed and proposed in literature and some of them could be embedded in the clustering process to perform an auto-refinement during the clustering itself (Manning et al., 2007, sec. 16.4.1).
In the sequel we propose a brief review of some of the most used indices[10], like *C-index* (Hubert and Schultz, 1976), *Goodman-Kruskal index* (Goodman and Kruskal, 1954), *Dunn index* (Dunn, 1974) and *Davies-Bouldin index* (Davies and Bouldin, 1979).

3.3.3.1 C-index

The *C-index* C is defined as

$$C = \frac{S - S_{\min}}{S_{\max} - S_{\min}} \tag{3.13}$$

where S, $S_{\min}$, $S_{\max}$ are computed as follows. Assume that p is the number of all pairs of samples for which both samples are located in the same cluster. Then S is the sum of distances between samples in those p pairs. Let P the number of all possible pairs of samples in the dataset. Ordering those P pairs by distance we can select p pairs with smallest and p pairs with largest distances between samples. We indicate with $S_{\min}$ the sum of the p smallest distances and with $S_{\max}$ the sum of the p largest distances.

[10]For more information about these indices and other ones, you could start from the bibliography (Bolshakova and Azuaje, 2005; Günter and Bunke, 2003; Halkidi et al., 2001; Zaïane et al., 2002).

From this formula it follows that the numerator will be small if pairs of samples with small distances are in the same cluster; thus, the smaller the value of C, the better the clusters. The denominator serves the purpose of normalization, causing $C \in [0, 1]$.

3.3.3.2 Goodman-Kruskal index

For calculating the Goodman-Kruskal (GK) index, all *quadruples* of input patterns in the dataset $\mathcal{X}$ are considered and the distance of the patterns are associated with the cluster membership.
Let $\delta(\cdot)$ be the distance between two objects (a and b, or c and d) in $\mathcal{X}$; a quadruple is called *concordant* if one of the following conditions is verified

$$\delta(a, b) < \delta(c, d) \tag{3.14}$$

where a and b are samples from the same cluster and c and d samples are from different clusters.
By contrast, a quadruple is called *discordant* if one of the following conditions is verified

$$\delta(a, b) < \delta(c, d) \tag{3.15}$$

where a and b are samples from different clusters and c and d samples are from the same cluster.
A good partition have many concordant and few discordant quadruples. Let N_c and N_d denote the number of concordant and discordant quadruples, respectively. Then the GK index is defined as

$$GK = \frac{N_c - N_d}{N_c + N_d} \tag{3.16}$$

The larger the GK value, the better the partition.

3.3.3.3 Dunn index

The Dunn index attempts to identify "compact and well separated clusters". The index is defined by the following equation for a specific number nc of clusters

$$D_{nc} = \min_{i=1..nc} \left\{ \min_{j=i+1..nc} \left[\frac{\mathrm{d}(\omega_i, \omega_j)}{\max_{k=1..nc} \mathrm{diam}(\omega_k)} \right] \right\} \tag{3.17}$$

where $\mathrm{d}(\cdot)$ and $\mathrm{diam}(\cdot)$ are defined as follows. Let δ be the dissimilarity function between two objects, $\mathrm{d}(\cdot)$ is the dissimilarity function between two clusters defined as

$$\mathrm{d}(\omega_i, \omega_j) = \min_{x \in \omega_i, y \in \omega_j} \delta(x, y) \tag{3.18}$$

and $\mathrm{diam}(\cdot)$ is the diameter of a cluster, defined as

$$\mathrm{diam}(\omega) = \max_{x,y \in \omega} \delta(x, y) \tag{3.19}$$

It is clear that $D \in [0, +\infty]$ with large values of D indicating the presence of compact and well separated clusters, so good clusterings. The drawback of this index is the high sensibility to the noise, so other more robust *Dunn-like* indices were developed across the years (Halkidi et al., 2001).

3.3.3.4 Daviese-Bouldin index

The Davies-Bouldin (DB) index is defined as follows

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{j=1,\cdots,k;j\neq i} (d_{ij}) \tag{3.20}$$

where

$$d_{ij} = \frac{\sigma_i + \sigma j}{\delta(o_i, o_j)}$$

In the formula, k is the number of clusters, σ_i is the average distance of all patterns in cluster i to their cluster center o_i, and $\delta(o_i, o_j)$ is the distance between cluster centers o_i and o_j. Hence, d_{ij} is small if clusters i and j are compact and their centers are far away from each other. Consequently, the lower the DB value, the better the clustering. It is easy to see that $DB \in [0, +\infty]$.

3.4 Estimating the number of clusters

The number k of desired clusters is an input parameter for the majority of clustering algorithms and as we stated in subsection 3.2.1, estimating k is often one of the most difficult tasks in the clustering process. Often k is nothing more than a good guess based on experience or domain knowledge and estimating it becomes very hard when we tackle totally new domains never treated before.
A first solution is to develop hierarchical clustering algorithms which do not use the achievement of the k clusters as stopping criterion and then examine the clustering results to understand what is the more appropriate number of clusters for the subdivision of the input dataset. But these "heuristics" have to be developed compatibly with the theoretical model which the algorithm is based on and so the process of "understanding the results" could change according to the hierarchical algorithm developed.
In reality, due to the fact that the estimation of the number of clusters is part of the evaluation of clustering, we could use the relative criteria to estimate the number of clusters, running several times the same algorithm with different parameters settings and then using an index to evaluate the quality of each result set. For instance, in the case we use the *C-index*, the lowest value obtained indicates the better parameter settings and so the best approximation or the exact value for the number of clusters.

3.5 More on results assessment

One type of result assessment is the already discussed *evaluation of clustering results*. Other types of results assessment are the *interpretation of clusters* and the *cluster visualization*. As in the case of the clustering evaluation, both of them usually occur at the end of the clustering process (see section 1.2).
The interpretability heavily depends on the adopted technique; for example, k-means and k-medoids algorithms produce clusters that are interpreted as dense areas around centroid or medoids and, therefore, score well.
The visualization of clusters becomes an hard task when we deal with high dimensional datasets, because human beings are not able to visually interpret shapes in more than three dimensions. Nowadays the high dimensional datasets are the standard in data mining application, so the need of instruments which allow us to easily visualize multi-dimensional clusters has grown in recent years. *Star Coordinates* seems a promising approach (Kandogan, 2000, 2001). Current experience with Star Coordinates on a number of datasets indicates that it is a viable approach to gain insight into multi-dimensional data.

3.6 High dimensional data and noise

Another difficulty we have to face when dealing with clustering is the dimensionality of data. The objects could be described by hundreds of attributes and this results in high dimensional datasets. In clustering, the overwhelming problem of high dimensionality presents a dual aspect. First, the presence of irrelevant attributes eliminates any hope on clustering tendency, because such features cause the algorithm search for clusters where there are no ones. This also happens with low dimensional data, but the likelihood of presence of irrelevant features and their number grow with dimension. The second problem is the so called *Curse of dimensionality*, which is a loose way of speaking about a lack of data separation in high dimensional space; this is why the distance between a point and its nearest neighbor becomes similar and similar to the distance to the majority of points as the dimensionality grows.[11] This effect starts to be severe for dimensions greater than $15/20$, therefore construction of clusters using proximity measures is doubtful in such situations.
There are two traditional ways to tackle the problem of high dimensionality. The first one consists in a variety of techniques to perform a *dimensionality reduction* before the clustering process, so we can work on a dataset ideally equivalent to the original one but with a lower dimensionality (and sometimes with a lower level of noise). The second way is known as *subspace clustering*, which is a spe-

[11] The *curse of dimensionality* is a term coined by Bellman (1961) to describe the problem caused by the exponential increase in volume associated with adding extra dimensions to a mathematical space. For example, 100 evenly-spaced sample points suffice to sample a unit interval with no more than 0.01 distance between points; an equivalent sampling of a 10-dimensional unit hypercube with a lattice with a spacing of 0.01 between adjacent points would require 10^{20} sample points: thus, in some sense, the 10-dimensional hypercube can be said to be a factor of 10^{18} "larger" than the unit interval.

cial class of clustering algorithms that try to circumvent high dimensionality by building clusters in appropriate subspaces of the original feature space.

3.6.1 Dimensionality reduction

Techniques for reduction of the dimensionality in datasets could be divided in three classes: *feature extraction* (also known as *feature transformation*), *feature selection* and, more recently, *feature clustering*. *Feature extraction* consists in applying a mapping from the multidimensional space to a space of fewer dimensions. This means that the original feature space is transformed by creating new features from combinations of the original ones. *Feature selection* methods, indeed, select only the most relevant of the dimensions from a dataset to reveal groups of objects that are similar on only a subset of their attributes. *Feature clustering* is a more recent class of methods for the reduction of dimensionality. It consists in performing the clustering of the feature set on a per-objects basis, i.e. it is a clustering of the transposed data matrix. The results of this process are clusters of features and the final goal of the feature clustering techniques is to use the feature clusters as new features for the clustering process, reducing drastically the dimensionality.

3.6.1.1 Feature extraction

Feature extraction is commonly used on high dimensional datasets. These methods include techniques such as PCA (Bishop, 2006, sec. 12.1), Singular Value Decomposition (SVD) or Sammon's non linear mapping (Sammon, 1969). The transformation generally preserves the original, relative distances between objects. In this way, they summarize the dataset by creating linear combinations of the attributes, and hopefully, uncover the latent structure of data.
Feature extraction is often a preprocessing step, allowing the clustering algorithm to use just a few of the newly created features. A few clustering methods have incorporated the use of such transformations to identify important features and iteratively improve their clustering. While often very useful, these techniques do not actually remove any of the original attributes from consideration. Thus, information from irrelevant dimensions is preserved, making these techniques ineffective at revealing clusters when there are large numbers of irrelevant attributes that mask the clusters with a huge amount of noise. Another disadvantage of using combinations of attributes is that they are difficult to interpret, often making clustering results less useful. Because of these problems, feature extraction is best suited to datasets where most of the dimensions are relevant to the clustering process, but many are highly correlated or redundant.

3.6.1.2 Feature selection

Feature selection attempts to discover the attributes of a dataset that are most relevant to the data mining task at hand. It is a commonly used and powerful technique for reducing the dimensionality of a problem to more manageable levels.

Feature selection involves searching through various feature subsets and evaluating each of these subsets using some criterion. The most popular search strategies are greedy sequential searches through the feature space, either forward or backward. The evaluation criteria follow one of two basic models, the *wrapper model* and the *filter model*. The wrapper model techniques evaluate the dataset using the data mining algorithm that will ultimately be employed; thus, they "wrap" the selection procedure around the data mining algorithm. Algorithms based on the filter model examine intrinsic properties of the data to evaluate the feature subset prior to data mining. Much effort in feature selection has been directed at supervised learning. Feature selection methods for supervised learning rely on the evaluation criteria like accuracy and/or on class labels. As we already know, in the unsupervised case we have neither class labels nor universally accepted evaluation criteria, but there are a number of methods that successfully adapt feature selection to clustering (see Parsons et al. (2004) for major references). However, while quite successful on a lot of datasets, feature selection algorithms have difficulty when clusters are found in different subspaces.

3.6.1.3 Feature clustering

We find early approaches of the feature clustering in text mining applications (Pereira et al., 1993), since this application domain suffers the *curse of dimensionality* very much. As showed in Baker and McCallum (1998); Slonim and Tishby (2001), the feature clustering process is more effective than feature selection methods. At the same time, feature clustering avoids also one of the main drawbacks of the feature extraction methods: the amplification of noise due to the extraction of new features as combination of other features. In fact, in opposition to feature selection, feature clustering do not throw away any feature, but perform a clustering on the feature set and therefore it ideally keeps most relevant portion of information about all features. Furthermore, unlike feature extraction methods, feature clustering performs also a reduction of noise since a clustering algorithm can be also viewed as a compression algorithm (see subsection 1.1.1).
In Dhillon et al. (2003a) the authors achieve very good results exploiting an information theoretic framework to perform the feature clustering; their algorithm has good performance minimizing the intra-cluster divergence and simultaneously maximizing the inter-cluster divergence.

3.6.2 Subspace clustering

Subspace clustering is an extension of traditional clustering that seeks to find clusters in different subspaces within a dataset (Parsons et al., 2004). Subspace clustering algorithms localize the search for relevant dimensions, therefore they are able to find clusters that exist in multiple, possibly overlapping subspaces. Just as with feature selection, subspace clustering requires a search method and an evaluation criterion. In addition, subspace clustering must somehow limit the scope of the evaluation criteria in order to consider different subspaces for each different cluster. An infeasible approach to searching for subspaces might be a

"brute force" search through all possible subspaces: by means of cluster validation techniques, we can determine in what subspace we can find the best clustering results. The unfeasibility lies in the intractability of the subset generation problem. Thus, more sophisticated heuristics are needed and they are grouped in two major branches based on the search strategy of algorithms. *Top-down* algorithms find an initial clustering in the full set of dimensions and evaluate the subspaces of each cluster, iteratively improving the results. *Bottom-up* approaches find dense regions in low dimensional spaces and combine them to form clusters. We know the major objective of clustering is to find high quality clusters within a reasonable time. While all clustering methods organize instances of a dataset in groups , some require cluster to be discrete (Hard Clustering) and others allow overlapping clusters (Soft or Fuzzy Clustering).[12] Subspace clustering algorithms must also define the subset of the features that are relevant for each cluster; these subspaces are almost always allowed to overlap. Furthermore, the subspace algorithms must determine the dimensionality for subspaces; they need also to be scalable with respect to subspaces dimensionality.
There are a lot of subspace clustering algorithms both in the top-down class and in the bottom-up class. Further subdivision of these two major branches and details about the most important subspace algorithms can be found in Parsons et al. (2004).

3.7 Sparseness and missing values

Another wearisome problem in data clustering is the sparseness of data matrix, better known as the *missing data values* problem. This occurs when, for some reason, an object in a dataset is not completely described, i.e. the related vector has missing components. Clustering algorithms generally have no internal way to handle missing values, so there are a variety of additional techniques to tackle this obstacle.
This problem often occurs in scientific data mining where the objects are usually described by a series of measurement. In this case, missing values occur due to observing conditions, instrument sensitivity limitations, and other real-world considerations. Application fields commonly affected by this complication are astronomy, astrophysics, biology, chemistry, but also speech recognition, computer vision and others.

3.7.1 Types of missing data values

Missing data values are usually divided into three categories (Westin, 2004). If objects who have missing data are a random subset of the complete sample of objects, missing values are called *Missing Completely At Random (MCAR)*. Typical examples of MCAR are when a tube containing a blood sample of a study subject is broken by accident (such that the blood parameters cannot be measured). The reason for missingness is completely random, i.e. the probability that an

[12] See subsection 3.2.2.

observation is missing is not related to any other feature. Instead, if the probability that an observation is missing depends on information that is not observed, like the value of the observation itself, missing data are called *Missing Not At Random (MNAR)*. Missing values other than MCAR and MNAR are called *Missing At Random (MAR)* and the reason of missingness in this case is based on other observed features.

3.7.2 Handling missing values

There are three common alternatives to handling missing values. The first one consists in *deletion* of either the cases reporting missing values or the features which cause missing values. The second one consists in ignoring them (*marginalization*). The last technique is called *data imputation*, and values are estimated to fill in missing values.

3.7.2.1 Removing missing data values

The method is easy and the data used are the data given and nothing else. However, the resulting data set is often biased and it is not a good choice to throw away information when building a decision support system. Furthermore, the problem persists, since nothing is done to handle missing data in the future use of the decision support system either. Removing missing data can be done in several ways. We can *remove cases*, if the portion of objects which report missing values is not too large. On the other hand, if some specific features cause a very large number of missing values, we can consider to *remove features*; again, this choice is not feasible if the number of such features is too large or they are highly characterizing. A combination of these two policies for missing data values deletion can be used too. However removing cases and/or features are always not a good idea, because we throw away a bit of information.

3.7.2.2 Imputation

Data are said to be imputed when they are obtained by some estimation method in order to replace missing values. Imputed data cannot and should not be considered as reliable as the actually observed data. It is important to exercise caution when drawing critical conclusions from data that is partially imputed. Estimated data should be flagged where possible to avoid drawing unwarranted conclusions. Despite this, data imputation is still commonly used without any mechanism to point to the imputed values, to warn about their unreliability. One approach is to replace all missing values with the observed mean for that feature (also known as the "row average" method in DNA microarray analysis (Wagstaff, 2004, sec. 2)). Another method is to model the observed values and select one according to the true distribution (if it is known). A more sophisticated approach is to infer the value of the missing feature by computing the Pearson correlation of each object with all other objects based on the known features and predict the

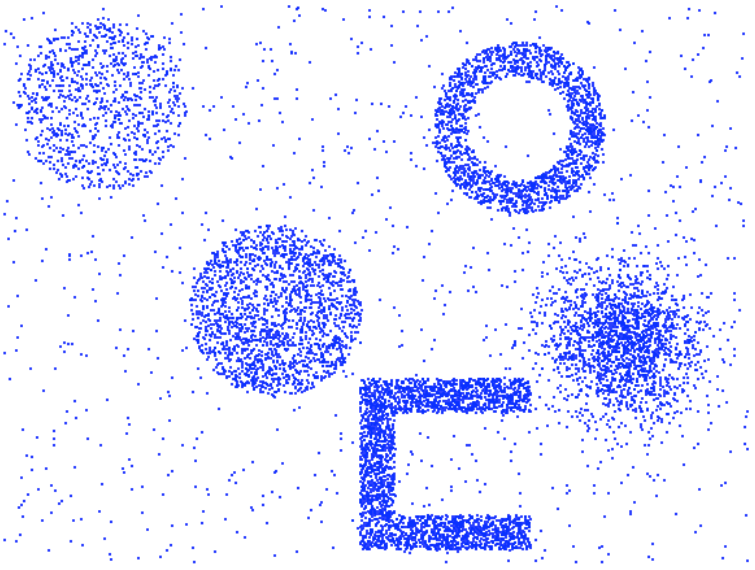


Figure 3.4: *In this example we can see five clusters with a low level of outliers randomly distributed in data space. (Pei and Zaïane, 2006, fig. 13)*

unknown value by proportionately combining the other objects' feature values (Banerjee et al., 2004a, sec. 4.2).

The imputation is usually a preprocessing step which we perform to transform the input data matrix before using it to feed the clustering algorithm. Several attempts to embed the preprocessing step in the clustering methods are done, like the *Preprocessing perceptron* (Westin, 2004). However, each of these methods suffers from an inability to discount imputed values due to their lack of full reliability.

3.7.2.3 Marginalization

The marginalization does not create any new data values and then could be a better solution. The missing values are simply ignored, but the clustering algorithms have to be designed in a way such that the marginalization does not nullify the clustering results. Although most of the effort in marginalization has focused on supervised methods, there are also several works to use marginalization in unsupervised cases. A remarkable approach following this strategy is the *clustering with soft constraints* (Wagstaff, 2004; Wagstaff and Laidler, 2005). This new approach divides the data features into *observed features*, which are known for all features, and *constraining features*, which contain missing values, and then it generates a set of constraints based on the known values for the constraining features. These new constraints are then combined with regular observed features in the clustering algorithm.

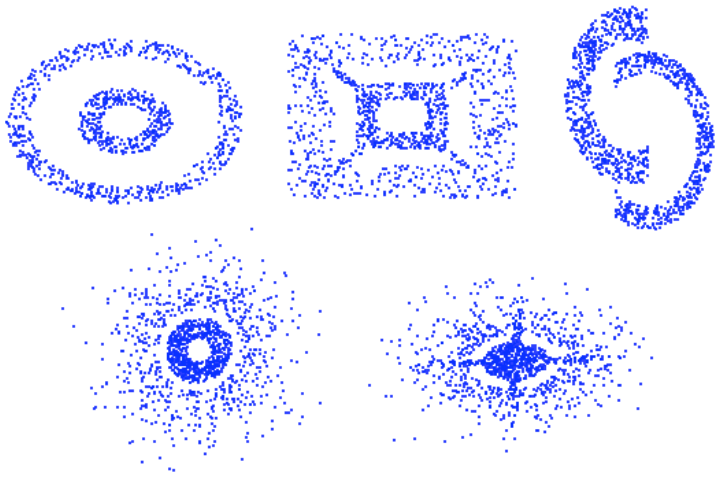


Figure 3.5: *An example of clusters with arbitrary shapes. (Pei and Zaïane, 2006, fig. 11)*

3.8 Outliers handling

In the machine learning and data mining context we call *outliers*[13] those points which do not belong to any cluster, i.e. outliers are the points which result completely foreign to the task at hand. For example, if we have to organize a set of plants in groups, an outlier can be a point which represent a mineral or an animal. Also, a point that originally would have had to belong to a cluster, could have become an outlier due to a huge amount of noise.
In general, clustering techniques do not distinguish between the outliers and other points and they tend to distribute the outliers among the clusters.
There are multiple ways of how outliers can be handled. If a data preprocessing phase is present, it usually takes care of outliers. Powerful techniques to detect outliers rely on unsupervised learning methods, such as the *novelty detection*. Once we have identified the outliers, we can remove them before starting the clustering process. An alternative to deletion is to correct, where possible, the outliers points, like proposed in Lekadir et al. (2007).
Certain algorithms have specific features for outliers handling, like some subspace clustering algorithms which are able to eliminate the subspaces with low coverage (Berkhin, 2002). Certain other algorithms are robust with respect to those outliers generated by a surplus of noise, like algorithms which adopt the feature clustering as a preprocessing step for dimensionality reduction, which performs also noise reduction.

[13]Outliers are often called *points of noise* or simply *noise* in machine learning and data mining literature. In this thesis we will always use the term outliers to indicate such points, whereas we call noise the disturbance on feature values (see section 3.6).

3.9 Clusters of arbitrary shape

Most of the clustering algorithms (both flat and hierarchical) are limited to clusters of convex shapes. Early approaches to handling clusters of arbitrary shapes came with some agglomerative hierarchical algorithms like CURE (Guha et al., 1998) or CHAMELEON (Karypis et al., 1999). Recent approaches include a generic clustering framework that approximates arbitrary shape clusters using the unions of small convex polygons (Choo et al., 2007) and the use of density estimation techniques (Jiang et al., 2007). However, arbitrary shape clusters handling is still an open and scarcely explored theme.

3.10 Conclusion

In this chapter we treated clustering in depth through a formal problem statement and a brief review of a categorization of the clustering algorithms. We have also reviewed the majority of the clustering issues. Among them, the problems we will focus on are the *high dimensionality* and the *missing values*. In addition, we will also take into account the *outliers handling* and *arbitrary shape clustering*. To address these problems we will introduce two clustering techniques at the state of the art. In the chapter 4 we will present some of the most widespread clustering algorithms, in order to compare them with aforementioned techniques discussed in the subsequent chapters of this thesis.

CHAPTER

FOUR

Previous works on clustering

«An algorithm must be seen to be believed.»

DONALD ERVIN KNUTH
(KNUTH, 1968, SEC. 1.1)

In the first part of the chapter we present a subset of the classical clustering algorithms proposed over the past years. We choose to expatiate about the most popular algorithms in the macro-category of flat clustering algorithms (see subsection 3.2.2). The "classic" clustering algorithms considered in the sequel are the already mentioned *K-means* (MacQueen, 1967) and the *Expectation-Maximization (EM)* algorithm (Dempster et al., 1977). The former is by far the most widespread clustering algorithm across multiple application domains; the latter is also very popular and it often serves as basic tool for several clustering algorithms. We also mention some variations of the K-means algorithm based on divergence functions other than euclidean distance. For each of them we present the theoretical background which they are based on and the relative advantages and disadvantages.
The second part of this chapter is reserved to a brief review of the few kernel methods for clustering. The kernel approach became famous thanks to the introduction of the *Support Vector Machines (SVMs)* (Vapnik, 1995) and is much more widespread in the supervised learning. Anyway, a small number of attempts to develop clustering algorithms which use the kernel approach was made and they often are extensions of well-known clustering techniques, like the "kernelized" version of the K-means (Dhillon et al., 2005b, 2004).
The first part of the chapter serves as term of comparison with the first clustering technique we will introduce in the sequel of this thesis: the *Minimum Bregman Information (MBI) principle for Co-clustering* (Banerjee et al., 2007). In the second

part of the chapter, we briefly review some kernel methods for clustering in order to point out the differences with the second clustering algorithm which we propose in the sequel of this thesis: the *Support Vector Clustering (SVC)* (Ben-Hur et al., 2001), based on the SVM formalism.

4.1 K-means

K-means[1] (MacQueen, 1967) is the most important flat clustering algorithm. Let us consider the dataset $\mathcal{X} \subseteq \mathbb{R}^d$. The objective function of K-means is to minimize the average squared distance of objects from their cluster centers, where a cluster center is defined as the mean or *centroid* $\vec{\mu}$ of the objects in a cluster $\mathcal{C}$:

$$\vec{\mu}(\mathcal{C}) = \frac{1}{\mid \mathcal{C} \mid} \sum_{\vec{x} \in \mathcal{C}} \vec{x} \tag{4.1}$$

The ideal cluster in K-means is a sphere with the centroid as its center of gravity. Ideally, the clusters should not overlap. A measure of how well the centroids represent the members of their clusters is the *Residual Sum of Squares (RSS)*, the squared distance of each vector from its centroid summed over all vectors

$$\text{RSS}_i = \sum_{\vec{x} \in \mathcal{C}_i} \|\vec{x} - \vec{\mu}(\mathcal{C}_i)\|^2 = \sum_{x \in \mathcal{C}_i} \sum_{m=1}^{d} (x_m - \mu_m(\mathcal{C}_i))^2 \tag{4.2}$$

$$\text{RSS} = \sum_{i=1}^{K} \text{RSS}_i \tag{4.3}$$

where x_m and $\mu_m(C_i)$ are the m^{th} components of $\vec{x}$ and $\vec{\mu}(C_i)$, respectively. We also recall that d is the dimension of the data space, i.e. the number of features. RSS is the objective function in K-means and the goal is to minimize it.[2] Since n, the number of input patterns, is fixed, minimizing RSS in equivalent to minimizing the average squared distance, a measure of how well centroids represent their objects.
K-means can start with selecting as initial clusters centers K randomly chosen objects, namely the *seeds*. It then moves the cluster centers around in space in order to minimize RSS. This is done iteratively by repeating two steps until a stopping criterion is met

(i) reassigning objects to the cluster with closest centroid

(ii) recomputing each centroid based on the current members of its cluster.

We can use one of the following termination conditions as stopping criterion

[1]The name comes from representing each of k clusters by the mean of its points, the so called *centroid*.

[2]This combinatorial problem is also known as the *minimum sum-of-squares* and is known to be NP-hard (Brucker, 1978; Merz, 2003).

- A fixed number of iterations I has been completed. This condition limits the runtime of the algorithm, but in some cases the quality of the clustering will be poor due to an insufficient number of iterations.
- Assignment of objects to clusters (the partitioning function γ) does not change between iterations. This produce a good clustering (expect in cases of poor local minimum), but the runtime may be unacceptably long.
- Centroids $\vec{\mu}_i$ do not change between iterations. This is equivalent to γ not changing.
- Terminate when RSS falls below a pre-estabilished threshold. This criterion makes sure that the clustering is of a desired quality after termination. In practice, we need to combine it with a bound on the number of iterations to guarantee the termination.

The K-means is shown to converge by proving that RSS monotonically decreases in each iteration. RSS decreases both in the *reassignment step* since each vector is assigned to the closest centroid and in the *recomputation step* because the new centroid is the vector $\vec{v}$ for which RSS_i reaches its minimum. Since there is only a finite set of possible clusterings, a monotonically decreasing algorithm will eventually arrive at a (local) minimum. The only care we have to take is the assignment of objects to the cluster with lowest index[3] among those having equidistant centroids, to avoid the algorithm cycling for ever in a loop of clusterings that have the same cost.
This proves the convergence of the K-means, but there is unfortunately no guarantee that a *global minimum* will be reached.

4.1.1 Computational complexity

Most of the time is spent on computing the vector distances. One such operation costs $O(d)$. The reassignment step computes $O(Kd)$ distances, so its overall complexity is $O(Knd)$. In the recomputation step, each vector gets added to a centroid once, so the complexity of this step is $O(nd)$. For a fixed number of iterations I, the overall complexity is therefore $O(IKnd)$. So, the K-means complexity is linear in all relevant factors: iterations, number of clusters, number of vectors and dimensionality of space.
There is one subtlety in the preceding argument. Even a linear algorithm can be quite slow if one of the arguments is large, and d could be very large in some application domain, such as document clustering. High dimensionality is not a problem for computing distance of two objects. Their vector are sparse, so that only small fraction of the theoretically possible d componentwise differences need to be computed. Centroids, however, are dense since they pool all terms that occur in any of the patterns of their clusters. As a result, distance computation is time consuming in a naive implementation of K-means, but there are a variety

[3] Or, equivalently, the highest index, or another unambiguous and constant priority order.

of heuristics for making centroid-object similarities as fast to compute as object-object similarities. Another way to address this efficiency problem is to use K-medoids algorithms. K-medoids algorithms are a variant of K-means ones; they computes medoids instead of centroids as cluster centers. A medoid of a cluster is the vector closest to the centroid. Since medoids are sparse vectors, distance computations are fast.

4.1.2 Advantages and disadvantages

The popularity of K-means is well deserved, since it is easily understood, easily implemented, and based on the firm foundation of analysis of variance. Furthermore, K-means is fast.
However, the K-means algorithm have also some drawbacks. First, as stated in the previous section, the computed local optimum may be quite different from the global one. This problem is amplified by the presence of *outliers* points. K-means have no way to detect and manage such points. Frequently, if an outlier is chosen as initial seed, no other vector will be assigned to it during subsequent iterations and we end up with a *singleton cluster* even though there is probably a clustering with lower RSS. This is why the K-means results highly depend on the initial guess of centroids. Therefore an heuristic strategy is necessary to select high quality seeds. Generally such heuristics embed a behavior to avoid outliers in the process of seed selection. Finally, the problem of estimating the number of clusters holds, even though there exist some heuristics to face this problem (Manning et al., 2007, sec. 16.4.1). A K-means modification, X-means, with built-in ability for estimating the right number of clusters also exists (Pelleg and Moore, 2000).

4.1.3 K-means variations

Recently it has been proved that many results attributed to the classical K-means with squared euclidean distance can be extended to many other distance-like functions, like the KL-divergence, also known as *relative entropy*; more generally, all distance functions based on Bregman[4] and Csiszar divergences can be used (Banerjee et al., 2005c; Teboulle et al., 2005).

4.2 Expectation-Maximization

The EM algorithm fall within a subcategory of the flat clustering algorithms, called *Model-based clustering*. The model-based clustering assumes that data were generated by a model and then tries to recover the original model from the data. This model then defines clusters and the cluster membership of data.

[4]The Bregman framework will be presented in the sequel as theoretical background for the *MBI principle for Co-clustering* (Banerjee et al., 2007).

Algorithm 1 The K-means algorithm

procedure KMEANS($\mathcal{X}, K$)
  $\{\vec{s}_1, \vec{s}_2, \cdots, \vec{s}_k\} \leftarrow SelectRandomSeeds(K, \mathcal{X})$
  for $i \leftarrow 1, K$ **do**
    $\vec{\mu}(\mathcal{C}_i) \leftarrow \vec{s}_i$
  end for
  repeat
    $\min_{\|\vec{x}_n - \vec{\mu}(\mathcal{C}_k)\|} \mathcal{C}_k = \mathcal{C}_k \cup \{\vec{x}_n\}$
    for all $\mathcal{C}_k$ **do**
      $\vec{\mu}(\mathcal{C}_k) = \frac{1}{|\mathcal{C}_k|} \sum_{\vec{x} \in \mathcal{C}_k} \vec{x}$
    end for
  until stopping criterion is met
end procedure

4.2.1 Maximum Likelihood

A commonly used criterion for selecting a model is the *Maximum Likelihood* criterion, which is also the one used in the development of the EM algorithm. A brief recall of the Maximum Likelihood problem follows.
We have a density function $p(x|\Theta)$ which is governed by the set of parameter Θ, i.e. the *model*. We also have a data set $\mathcal{X} = \{x_1, x_2, \cdots, x_N\}$, that we suppose to be drawn by the probability distribution p. We assume that the data vectors of $\mathcal{X}$ are *i.i.d.* with distribution p. Therefore, the resulting density for the samples is

$$p(\mathcal{X}|\Theta) = \prod_{i=1}^{N} p(x_i|\Theta) = \mathcal{L}(\mathcal{X}|\Theta) \tag{4.4}$$

The function $\mathcal{L}(\mathcal{X}|\Theta)$ is called *likelihood function*. We can see the likelihood as a function of the model Θ where the dataset $\mathcal{X}$ is fixed.
We can now state the problem formally.

Problem 4.1 (Maximum Likelihood) *The Maximum Likelihood criterion consists in finding the model Θ^* that maximizes the likelihood $\mathcal{L}(\mathcal{X}|\Theta)$ of generating the data, that is*

$$\Theta^* = \arg\max_{\Theta} \mathcal{L}(\mathcal{X}|\Theta) \tag{4.5}$$

We often maximize its logarithm,[5] $\log \mathcal{L}(\mathcal{X}|\Theta)$, since it is easier, so we can define the *log likelihood function* as follows

$$\hat{\mathcal{L}}(\mathcal{X}|\Theta) = \log \mathcal{L}(\mathcal{X}|\Theta)) = \log \prod_{i=1}^{N} p(x_i|\Theta) = \sum_{i=1}^{N} \log p(x_i|\Theta) \tag{4.6}$$

and then we can reformulate the problem 4.1.

[5]Hereafter, we assume that both $\log$ and $\ln$ refer to the natural logarithm. All other logarithm bases will be explicitly specified.

Problem 4.2 (Maximum Log-likelihood) *The Maximum Log-likelihood criterion consists in finding the model Θ^* that maximizes the log likelihood $\hat{\mathcal{L}}(\mathcal{X}|\Theta)$ of generating the data, that is*

$$\Theta^* = \arg\max_{\Theta} \log \hat{\mathcal{L}}(\mathcal{X}|\Theta) \tag{4.7}$$

The (log) likelihood function can be used to measure the goodness of the clustering. Given two clusterings with the same number of clusters, we prefer the one with higher (log) likelihood.
Due the fact that the log-likelihood is easier to compute, in literature it is often called simply likelihood, understanding the "log" variant. We adopt the same assumptions from now on: the $\mathcal{L}(\mathcal{X}|\Theta)$ function will indicate the log-likelihood function.

4.2.2 The algorithm

The Expectation-Maximization is an iterative algorithm that finds the maximum likelihood estimation of the parameters of an underlying distribution from a given dataset. There are two main applications of the EM algorithm. The former occurs when the data has missing values, due to the limitations of the observation process. The latter occurs when optimizing the likelihood function is analytically hard and the likelihood function can be simplified by assuming the existence of additional *missing* (or *hidden*) parameters. The latter application is commonly used in clustering.
As we stated in previous section, we assume that the data set $\mathcal{X}$ is generated by a probability distribution p. We call $\mathcal{X}$ the *incomplete data*. We also assume that a complete dataset $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$ exists and finally assume a joint density function

$$p(z|\Theta) = p(x, y|\Theta) = p(y|x, \Theta)p(x|\Theta) \tag{4.8}$$

This density often arises from the marginal density function $p(x|\Theta)$ and the assumption of hidden variables and parameter value guesses. In other cases (e.g. missing data values in samples of a distribution), we have to assume a joint relationship between the missing and the observed values.
With this new density function, we can define a new likelihood function

$$\mathcal{L}(\mathcal{Z}|\Theta) = \mathcal{L}(\mathcal{X}, \mathcal{Y}|\Theta) = p(\mathcal{X}, \mathcal{Y}|\Theta) \tag{4.9}$$

called the *complete data likelihood*.
This function is a random variable, since the missing information $\mathcal{Y}$ is unknown, random, and presumably governed by an underlying distribution. Hence we can state

$$\mathcal{L}(\mathcal{X}, \mathcal{Y}|\Theta) = \mathrm{h}_{\mathcal{X},\Theta}(\mathcal{Y}) \tag{4.10}$$

where $\mathcal{X}$ and Θ are constants, $\mathcal{Y}$ is a random variable and $\mathrm{h}(\cdot)$ is an unknown function.
The original likelihood function $\mathcal{L}(\mathcal{X}|\Theta)$ is called *incomplete data likelihood function*.

4.2.2.1 Expectation step

The first step of the EM algorithm finds the expectation E of the complete data likelihood with respect to the unknown data $\mathcal{Y}$ given the observed data $\mathcal{X}$ and the current parameter estimates Θ. We define

$$Q(\Theta, \Theta^{(i-1)}) = E[\mathcal{L}(\mathcal{Z}|\Theta)|\mathcal{X}, \Theta^{(i-1)}] \tag{4.11}$$

where $\Theta^{(i-1)}$ are the current parameter estimates we used to evaluate the expectation and Θ are the new parameters we optimize to increase Q.
We observe that $\mathcal{X}$ and $\Theta^{(i-1)}$ are constants, whereas Θ is the variable we want to tune and $\mathcal{Y}$ is a random variable governed by the distribution $f(y|\mathcal{X}, \Theta^{(i-1)})$. So, the right side of Equation 4.11 can be rewritten as

$$E[\mathcal{L}(\mathcal{Z}|\Theta)|\mathcal{X}, \Theta^{(i-1)}] = \int_{y \in \mathcal{Y}} \log p(\mathcal{X}, y|\Theta) f(y|\mathcal{X}, \Theta^{(i-1)}) dy \tag{4.12}$$

where $f(y|\mathcal{X}, \Theta^{(i-1)})$ is the marginal distribution of the unobserved data and is dependent on $\mathcal{X}$ and on the current parameters $\Theta^{(i-1)}$. The evaluation of the Equation 4.12 is the proper *Expecation step* or *E-step*.

4.2.2.2 Maximization step

The second step (the *Maximization step* or *M-step*) of the EM algorithm is to maximize the expectation that has been computed in the first step. We find

$$\Theta^{(i)} = \arg\max_{\Theta} Q(\Theta, \Theta^{(i-1)}) \tag{4.13}$$

The two steps of the EM algorithm are repeated until the algorithm converges. Each iteration is guaranteed to increase the likelihood and the algorithm is guaranteed to converge to a local maximum of the likelihood function. The EM, in practice, converges after few iterations, a property which caused the popularity of this clustering scheme in machine learning research community.

4.2.3 Advantages and disadvantages

As we stated above, the main property of the EM is the ensured convergence in few iterations. Anyway, the EM does not reach a global maximum for the likelihood function, but it converges to a local maximum depending on starting values. There are a variety of heuristic approaches for escaping a local maximum such as using several different random initial estimates or applying simulated annealing (Lavielle and Moulines, 1997).
EM is particularly useful when maximum likelihood estimation of a complete data model is easy. If closed-form estimators exist, the M-step is often trivial. A classic example is maximum likelihood estimation of a finite mixture of Gaussians, where each component of the mixture can be estimated trivially if the mixing distribution is known.

Besides, as already stated, the EM acts as a general framework for developing other algorithms. For instance, we can derive the K-means algorithm as a particular limit of the EM for Gaussian mixtures (Bishop, 2006, sec. 9.3.2).

4.3 Kernel Methods for Clustering

As we stated in the beginning of this chapter, there are few clustering algorithms based on kernels. The largest part of these algorithms is an extension of classical algorithms in which we substitute the measurement of vector distances in data space with the same distances computed in a higher dimensional features space by means of a Mercer kernel (see subsection 2.6.1). We call this process *metric kernelization* (Camastra, 2004, sec. 2.7) and it was applied to create kernel extensions of famous algorithms, like *Nearest Neighbor* and *Fuzzy C-means* (Camastra, 2004, sec. 7.2). There is also a kernel version of *K-means* (Dhillon et al., 2005b, 2004) and a recent approach to a kernel-based graph clustering (Dhillon et al., 2005a).

4.3.1 Metric kernelization

Let $\phi : \mathbb{R}^d \rightarrow \mathcal{F}$ be a mapping from $\mathbb{R}^d$ to a higher dimensional feature space $\mathcal{F}$ and let $\vec{x}, \vec{y} \in \mathbb{R}^d$ be two points, we define the *kernelized* metric $d_K(\vec{x}, \vec{y})$ between $\phi(\vec{x})$ and $\phi(\vec{y})$ as

$$d_K(\vec{x}, \vec{y}) = \sqrt{K(\vec{x}, \vec{x}) - 2K(\vec{x}, \vec{y}) + K(\vec{y}, \vec{y})} \tag{4.14}$$

where $K(\cdot)$ is a Mercer kernel such that $K(\vec{x}, \vec{y}) = \phi(\vec{x}) \cdot \phi(\vec{y})$.

4.3.1.1 An example: kernelized K-means

To understand better the concept of metric kernelization, we present here the reformulation of the K-means. Let us redefine the centroid first (see Equation 4.1)

$$\vec{\mu}(\mathcal{C}) = \frac{1}{\mid \mathcal{C} \mid} \sum_{\vec{x} \in \mathcal{C}} \phi(\vec{x}) \tag{4.15}$$

Next, Equation 4.2 becomes

$$\text{RSS}_i = \sum_{\vec{x} \in \mathcal{C}_i} \| \phi(\vec{x}) - \vec{\mu}(\mathcal{C}_i) \|^2 \tag{4.16}$$

Using the relation Equation 4.14 we have

$$\text{RSS}_i = \sum_{\vec{x} \in \mathcal{C}_i} \phi(\vec{x})\phi(\vec{x}) - \frac{2 \sum_{\vec{y} \in \mathcal{C}} \phi(\vec{x})\phi(\vec{y})}{\mid \mathcal{C} \mid} + \frac{\sum_{\vec{y}, \vec{z} \in \mathcal{C}} \phi(\vec{y})\phi(\vec{z})}{\mid \mathcal{C} \mid^2} \tag{4.17}$$

Replacing the dot products in the feature space with a Mercer kernel, Equation 4.17 turns in

$$\text{RSS}_i = \sum_{\vec{x} \in \mathcal{C}_i} K(\vec{x}, \vec{x}) - \frac{2 \sum_{\vec{y} \in \mathcal{C}} K(\vec{x}, \vec{y})}{\mid \mathcal{C} \mid} + \frac{\sum_{\vec{y}, \vec{z} \in \mathcal{C}} K(\vec{y}, \vec{z})}{\mid \mathcal{C} \mid^2} \tag{4.18}$$

4.3.2 Advantages and disadvantages

The clustering algorithms which rely on *metric kernelization* are a first attempt to face nonlinear separable clustering problems. However this technique presents some problems. First, the metric kernelization can be applied only to particular clustering algorithms. Furthermore, the convergence of a kernelized algorithm cannot be proved in all cases.[6] Finally, if we compare such algorithms with the state-of-art of kernel methods for supervised learning, i.e. the SVMs, we can notice that the SVMs provide us with a support vector description of the data. Hence, the metric kernelization alone often does not justify the increment of the computational complexity due to the employment of kernels to compute metrics.

4.4 Conclusion

In the first part of this chapter we reviewed the most famous flat clustering algorithms and reported the relative advantages and disadvantages. The algorithms presented are the most similar ones to the novel clustering framework based on the Minimum Bregman Information (MBI) principle which we will exploit in particular for the co-clustering. In the second half of the chapter we saw a first attempt to exploit the kernel approach in clustering. In the sequel we will introduce a kernel method for clustering based on the Support Vector Machine (SVM) formalism. So, this chapter aims at providing a comparison term for the methodologies that we will introduce in the subsequent chapters. We will show how these innovative techniques solve some important clustering issues.

[6] The convergence of kernelized K-means analyzed above was proved (Dhillon et al., 2005b, sec. 2.2).

CHAPTER

FIVE

Minimum Bregman Information principle for Co-clustering

«You should call it entropy, for two reasons. In the first place your uncertainty function has been used in statistical mechanics under that name, so it already has a name. In the second place, and more important, no one really knows what entropy really is, so in a debate you will always have the advantage.[a]»

[a] Suggesting to Claude Shannon a name for his new uncertainty function.
JOHN VON NEUMANN
(MCIRVINE AND TRIBUS, 1971, P. 180)

In this chapter we introduce a very interesting result: the *Minimum Bregman Information (MBI) principle* for clustering, that simultaneously generalizes the *Maximum Entropy* principle for clustering, the *Standard Least Squares* principle for clustering and other ones (Banerjee et al., 2005c). In particular, we present the application of such a principle for the solution of a different and more general clustering problem, namely the *Co-clustering* (Banerjee et al., 2004a,b, 2007).
The co-clustering[1] is the simultaneous clustering of both the rows and the columns of a data matrix, i.e. the simultaneous clustering of both objects and attributes.

[1] Also known as *simultaneous clustering*, *bidimensional clustering*, *bi-clustering*, *block clustering*, *conjugate clustering*.

By means of the MBI principle and thanks to the co-clusteirng scheme, we achieve a twofold generalization of the classic hard clustering relocation scheme such as the one used by the K-means algorithm. First, we are no longer constrained to the squared Euclidean distance, but we have at our disposal a large set of distortion functions: the Bregman divergences. In the second place, we are able to choose what we want to cluster: data, features or both at the same time. Additionally, a soft-clustering formulation of both clustering and co-clustering problem via Bregman divergences is available (Banerjee et al., 2005b,c; Gupta and Chandola, 2006; Shafiei and Milios, 2006).
However, in this chapter we focus on hard clustering formulation. We first introduce the *Bregman divergences* which are the basis of the MBI principle, and formulate the classic one-way clustering. Then, we generalize the discussion to the co-clustering problem and show what are the advantages of the simultaneous clustering of both data and features alongside the advantages of using a larger class of loss functions. Finally, we conclude the chapter with some other applications of the Bregman co-clustering, such as the matrix approximation problem and the missing values prediction.

5.1 Towards the Bregman divergences

Several works over the past years have motivated the investigation for a more general class of distortion functions. In fact, the classic relocation scheme adopted by K-means (see section 4.1) has been employed also in other algorithms that use distortion functions other than the squared Euclidean one. For example, the *Linde-Buzo-Gray (LBG)* algorithm uses a K-means like strategy but adopts the Itakura-Saito distance (Linde et al., 1980). Another example is the Entropy-like K-means which is a K-means version that use the KL-divergence (also known as *relative entropy*) (Teboulle et al., 2005).
From the above examples, it is clear that the iterative relocation scheme is a successful approach to the clustering. Therefore, this has led to reveal what class of distortion functions was suitable with such a scheme, in order to have a more general clustering algorithm. Such functions are the *Bregman divergences*, which include also the aforementioned distances.

5.1.1 Affine sets and relative interior

Let us provide some preparatory definitions for the introduction of the Bregman divergences.
A set $C \subseteq \mathbb{R}^n$ is said to be *affine* if the line through any two distinct points in C lies in C (Boyd and Vandenberghe, 2004, chap. 2), i.e. if

$$\forall \vec{x}_1, \vec{x}_2 \in C \exists \theta \in \mathbb{R} \colon \theta \vec{x}_1 + (1-\theta)\vec{x}_2 \in C. \tag{5.1}$$

In other words, C contains the linear combination of any two points in C, provided the coefficients in the linear combination sum to one.

This idea can be generalized to more than two points. We refer to a point of the form $\theta_1\vec{x}_1, \theta_1\vec{x}_2, \cdots, \theta_1\vec{x}_p$ where $\theta_1 + \theta_2 + \cdots + \theta p = 1$, as an *affine combination* of the points $\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_p$. Using the induction from the definition of affine set, it can be shown that an affine set contains every affine combination of its points.
The set of all affine combinations of points in some set $C \subseteq \mathbb{R}^n$ is called the *affine hull* of C, and denoted aff(C)

$$\text{aff}(C) = \{\theta_1\vec{x}_1 + \theta_1\vec{x}_2 + \cdots + \theta_1\vec{x}_p \colon \vec{x}_1, \vec{x}_2, \cdots, \vec{x}_p \in C, \theta_1 + \theta_2 + \cdots + \theta p = 1\}.$$

The affine hull is the smallest affine set that contains C, in the following sense: if S is any affine set with $C \subseteq S$, then aff$(C) \subseteq S$.
Finally, we define the *relative interior* of the set C, denoted ri(C), as its interior[2] relative to aff(C)

$$\text{ri}(C) = \{\vec{x} \in C \colon B(\vec{x}, r) \cap \text{aff}(C) \subseteq C \text{ for some } r > 0\}, \tag{5.2}$$

where $B(\vec{x}, r)$ is the ball of radius r and center x (Boyd and Vandenberghe, 2004, chap. 2).

5.1.2 Bregman divergences

Let us define *Bregman divergences* (Bregman, 1967), which form a large class of well-behaved loss functions with a number of desirable properties.

Definition 5.1 (Bregman divergence) *Let ϕ be a real-valued convex function of Legendre type*[3] *defined on the convex set $\mathcal{S} \equiv dom(\phi) \subseteq \mathbb{R}^d$. The* **Bregman divergence** *$d_\phi : \mathcal{S} \times ri(\mathcal{S}) \to \mathbb{R}_+$ is defined as*

$$d_\phi(\vec{x}_1, \vec{x}_2) = \phi(\vec{x}_1) - \phi(\vec{x}_2) - \langle \vec{x}_1 - \vec{x}_2, \nabla\phi(\vec{x}_2)\rangle \tag{5.3}$$

where $\nabla\phi$ is the gradient of ϕ, $\langle\cdot\rangle$ is the dot product, and ri$(\mathcal{S})$ is the relative interior of $\mathcal{S}$.

Example 5.1 (Squared Euclidean Distance) Squared Euclidean distance *is perhaps the simplest and most widely used Bregman divergence. The underlying function $\phi(\vec{x}) = \langle \vec{x}, \vec{x}\rangle$ is strictly convex, differentiable in $\mathbb{R}^d$ and*

$$\begin{aligned} d_\phi(\vec{x}_1, \vec{x}_2) &= \langle \vec{x}_1, \vec{x}_1\rangle - \langle \vec{x}_2, \vec{x}_2\rangle - \langle \vec{x}_1 - \vec{x}_2, \nabla\phi(\vec{x}_2)\rangle = \\ &= \langle \vec{x}_1, \vec{x}_1\rangle - \langle \vec{x}_2, \vec{x}_2\rangle - \langle \vec{x}_1 - \vec{x}_2, 2\vec{x}_2\rangle = \\ &= \langle \vec{x}_1 - \vec{x}_2, \vec{x}_1 - \vec{x}_2\rangle = \|\vec{x}_1 - \vec{x}_2\|^2 \end{aligned} \tag{5.4}$$

[2]The interior of a set C consists of all points of C that are intuitively not on the "edge" of C (Boyd and Vandenberghe, 2004, app. A).

[3]A proper, closed, convex function ϕ is said to be of Legendre type if: (i) int(dom(ϕ)) is non-empty, (ii) ϕ is strictly convex and differentiable on int(dom(ϕ)), and (iii) $\forall z_b \in$ bd(dom(ϕ)), $\lim_{z \in \text{dom}(\phi) \to z_b} \|\nabla\phi(z)\| \to \infty$, where dom$(\phi)$ is the domain of the ϕ application, int(dom(ϕ)) is the interior of the domain of ϕ and bd(dom(ϕ)) is the boundary of the domain of ϕ (Banerjee et al., 2005c).

Domain	$\phi(x)$	$d_\phi(x, y)$	**Divergence**
$\mathbb{R}$	x^2	$(x-y)^2$	Squared loss
$\mathbb{R}_{++}$	$-\log x$	$\frac{x}{y} - \log(\frac{x}{y}) - 1$	Itakura-Saito
$\mathbb{R}^d$	$\|x\|^2$	$\|x-y\|^2$	Squared Euclidean
$\mathbb{R}^d$	$x^T A x$	$(x-y)^T A (x-y)$	Mahalanobis
d-Simplex	$\sum_{j=1}^d x_j \log_2 x_j$	$\sum_{j=1}^d x_j \log_2(\frac{x_j}{y_j})$	KL-divergence
$\mathbb{R}^d$	$\sum_{j=1}^d x_j \log x_j$	$\sum_{j=1}^d x_j \log(\frac{x_j}{y_j}) - \sum_{j=1}^d (x_j - y_j)$	I-divergence
$\mathbb{R}$	e^x	$e^x - e^y - (x-y)e^y$	Unnamed
$\mathbb{R}_+$	$x \log x$	$x \log(\frac{x}{y}) - (x-y)$	Unnamed

Table 5.1: *Bregman divergences generated from some convex functions.*

Example 5.2 (Relative Entropy) *Another widely used Bregman divergence is the KL-divergence, also know as* Relative Entropy. *Let $p(\cdot)$ and $q(\cdot)$ be two discrete probability distributions so that $\sum_{j=1}^d p_j = 1$ and $\sum_{j=1}^d q_j = 1$, the negative entropies $\phi(p) = \sum_{j=1}^d p_j \log_2 p_j$ and $\phi(q) = \sum_{j=1}^d q_j \log_2 q_j$ are convex functions. The corresponding Bregman divergence is*

$$\begin{aligned} d_\phi(p,q) &= \sum_{j=1}^d p_j \log_2 p_j - \sum_{j=1}^d q_j \log_2 q_j - \langle p-q, \nabla\phi(q)\rangle = \\ &= \sum_{j=1}^d p_j \log_2 p_j - \sum_{j=1}^d q_j \log_2 q_j - \sum_{j=1}^d (p_j - q_j)(\log_2 q_j - \log_2 e) = \\ &= \sum_{j=1}^d p_j \log_2 \left(\frac{p_j}{q_j}\right) - \log_2 e \sum_{j=1}^d (p_j - q_j) = \\ &= KL(p\|q) \end{aligned} \tag{5.5}$$

that is, the KL-divergence between the two distributions.

5.1.3 Bregman information

The dual formulation of the Shannon's rate distortion problem involves finding a coding scheme with a given rate, i.e. average number of bits per symbol, such that the expected distortion between the source random variable and the decoded random variable is minimized. The achieved distortion is called *distortion rate function*, i.e. the infimum distortion achievable for a given rate. Now consider a random variable X that takes values in a finite set $\mathcal{X} = \{\vec{x}_i\}_{i=1}^n \subset \mathcal{S} \subseteq \mathbb{R}^d$ (where $\mathcal{S}$ is convex) following a discrete probability measure ν. Let the distortion be measured by a Bregman divergence $d_\phi(\cdot)$. Consider a simple encoding scheme that represents the random variable by a constant vector $\vec{s} \in \text{ri}(\mathcal{S})$, i.e. the codebook size is one (equivalently, the rate is zero). The solution to the rate-distortion problem in this case is the trivial assignment. The corresponding distortion-rate function is given by

$$\mathbb{E}_\nu[d_\phi(X, \vec{s})] \tag{5.6}$$

that depends on the choice of the representative $\vec{s}$ and can be optimized by picking up the right representative. We call this optimal distortion-rate function the *Bregman Information (BI)* of the random variable X for the Bregman divergence d_ϕ and denote it by $I_\phi(X)$, i.e.

$$I_\phi(X) = \min_{\vec{s} \in \mathbf{ri}(\mathcal{S})} \mathbb{E}_\nu[d_\phi(X, \vec{s})] = \min_{\vec{s} \in \mathbf{ri}(\mathcal{S})} \sum_{i=1}^{n} \nu_i d_\phi(\vec{x}_i, \vec{s}) \tag{5.7}$$

The optimal vector $\vec{s}$ which achieves the minimal distortion will be called the *Bregman representative* or, simply the *representative* of X. The following proposition[4] states that such a representative always exists, is uniquely determined and *does not depend* on the choice of Bregman divergence. In fact, the minimizer is just the expectation of the random variable X.

Proposition 5.1 *Let X be a random variable that takes values in $\mathcal{X} = \{\vec{x}_i\}_{i=1}^n \subset \mathcal{S} \subseteq \mathbb{R}^d$ following a positive probability distribution measure ν such that $\mathbb{E}_\nu[X] \in ri(\mathcal{S})$.[5] Given a Bregman divergence $d_\phi : \mathcal{S} \times ri(\mathcal{S}) \to [0, \infty)$, the problem*

$$\min_{\vec{s} \in ri(\mathcal{S})} \mathbb{E}_\nu[d_\phi(X, \vec{s})] \tag{5.8}$$

has a unique solution given by $\vec{s}^* = \vec{\mu} = \mathbb{E}_\nu[X]$.

Using the proposition above, we can now give a more direct definition of the *Bregman Information (BI)*.

Definition 5.2 (Bregman Information) *Let X be a random variable that take values in $\mathcal{X} = \{\vec{x}_i\}_{i=1}^n \subset \mathcal{S} \subseteq \mathbb{R}^d$ following a positive probability distribution measure ν. Let $\mu = \mathbb{E}_\nu[X] = \sum_{i=1}^n \nu_i \vec{x}_i \in ri(\mathcal{S})$ and let $d_\phi \colon \mathcal{S} \times ri(\mathcal{S}) \to [0, \infty)$ be a Bregman divergence. Then the* **Bregman Information** *of X in terms of d_ϕ is defined as*

$$I_\phi(X) = \mathbb{E}_\nu[d_\phi(X, \vec{\mu})] = \sum_{i=1}^{n} \nu_i d_\phi(\vec{x}_i, \vec{\mu}) \tag{5.9}$$

Example 5.3 (Variance) *Let $\mathcal{X} = \{\vec{x}_i\}_{i=1}^n$ be a set in $\mathbb{R}^d$, and consider the uniform measure over $\mathcal{X}$, i.e. $\nu_i = 1/n$. The Bregman Information of X with squared Euclidean distance as Bregman divergence is actually the variance*

$$I_\phi(X) = \sum_{i=1}^{n} \nu_i d_\phi(\vec{x}_i, \vec{\mu}) = \frac{1}{n} \sum_{i=1}^{n} \|\vec{x}_i - \vec{\mu}\|^2 \tag{5.10}$$

where

$$\vec{\mu} = \sum_{i=1}^{n} \nu_i \vec{x}_i = \sum_{i=1}^{n} \frac{1}{n} \vec{x}_i$$

[4] A proof is available in Banerjee et al. (2005c).

[5] The assumption that $\mathbb{E}_\nu[X] \in \mathrm{ri}(\mathcal{S})$ is not restrictive since a violation can occur only when $\mathrm{co}(\mathcal{X}) \subset \mathrm{bd}(\mathcal{S})$, i.e. the entire convex hull of $\mathcal{X}$ is on the boundary of $\mathcal{S}$.

is the arithmetic mean.

Example 5.4 (Mutual Information) *By definition, the mutual information of $I(U;V)$ between two discrete random variables U and V with joint distribution $\{\{p(\vec{u}_i,\vec{v}_j\}_{i=1}^n\}_{j=1}^m$ is given by*

$$\begin{aligned} I(U;V) &= \sum_{i=1}^{n}\sum_{j=1}^{m} p(\vec{u}_i,\vec{v}_j)\log\frac{p(\vec{u}_i,\vec{v}_j)}{p(\vec{u}_i)p(\vec{v}_j)} = \\ &= \sum_{i=1}^{n} p(\vec{u}_i)\sum_{j=1}^{m} p(\vec{v}_j|\vec{u}_i)\log\frac{p(\vec{v}_j|\vec{u}_i)}{p(\vec{v}_j)} = \\ &= \sum_{i=1}^{n} p(\vec{u}_i)KL(p(V|\vec{u}_i)\|p(V)) \end{aligned} \tag{5.11}$$

Consider a random variable Z_u that takes values in the set of probability distributions $\mathcal{Z}_u = \{p(V|\vec{u}_i)\}_{i=1}^n$ following the probability measure $\{\nu_i\}_{i=1}^n = \{p(\vec{u}_i)\}_{i=1}^n$ over this set. The mean (distribution) of Z_u is given by

$$\vec{\mu} = \mathbb{E}_\nu[p(V|\vec{u})] = \sum_{i=1}^{n} p(\vec{u}_i)p(V|\vec{u}_i) = \sum_{i=1}^{n} p(\vec{u}_i,V) = p(V) \tag{5.12}$$

Hence,

$$I(U;V) = \sum_{i=1}^{n} \nu_i d_\phi(p(V|\vec{u}_i),\vec{\mu}) = I_\phi(Z_u) \tag{5.13}$$

that is, the mutual information is the Bregman Information of Z_u when d_ϕ is the KL-divergence. Similarly, for a random variable Z_v that takes values in the set of the probability distributions $\mathcal{Z}_v = \{p(U|\vec{v}_j\}_{j=1}^m$ following the probability measure $\{\nu_j\}_{j=1}^m = \{p(\vec{v}_j)\}_{j=1}^m$ over this set, one can show that $I(U;V) = I_\phi(Z_v)$.

5.2 Hard Clustering with Bregman divergences

Let X be a random variable that takes values in $\mathcal{X} = \{\vec{x}_i\}_{i=1}^n$ following the probability distribution ν. When X has a large Bregman Information, it may not suffice to encode X using a single representative since a lower quantization error may be desired. In such a situation, a natural goal is to split the set $\mathcal{X}$ into k disjoint partitions $\{\mathcal{X}_h\}_{h=1}^k$, each with its Bregman representative, such that a random variable M over the partition representatives serves as an appropriate quantization of X. Let $\mathcal{M} = \{\vec{\mu}_h\}_{h=1}^k$ denote the set of representatives, and $\pi = \{\pi_h\}_{h=1}^k$ with $\pi_h = \sum_{\vec{x}_i\in\mathcal{X}_h}\nu_i$ denote the induced probability measure on $\mathcal{M}$. Then the induced random variable M takes values in $\mathcal{M}$ following π.
The quality of the quantization M can be measured by the expected Bregman divergence between X and M, i.e. $\mathbb{E}_{X,M}[d_\phi(X,M)]$. Since M is a deterministic function of X, the expectation is actually over the distribution of X, so that

$$\begin{aligned}\mathbb{E}_X[d_\phi(X, M)] &= \sum_{h=1}^{k} \sum_{\vec{x}_i \in \mathcal{X}_h} \nu_i d_\phi(\vec{x}_i, \vec{\mu}_h) = \\ &= \sum_{h=1}^{k} \pi_h \sum_{\vec{x}_i \in \mathcal{X}_h} \frac{\nu_i}{\pi_h} d_\phi(\vec{x}_i, \vec{\mu}_h) = \\ &= \mathbb{E}_\pi[I_\phi(X_h)]\end{aligned} \tag{5.14}$$

where X_h is the random variable that takes values in the partition $\mathcal{X}_h$ following a probability distribution ν_i/π_h, and $I_\phi(X_h)$ is the Bregman Information of X_h. Thus, the quality of the quantization is equal to the expected Bregman Information of the partitions.
An alternative way of measuring the quality of the quantization M can be formulated from an information theoretic viewpoint. We can measure the quality of M by the loss in Bregman Information due to the quantization, i.e. by

$$I_\phi(X) - I_\phi(M) \tag{5.15}$$

For $k = n$, the best choice is of course $M = X$ with no loss in Bregman Information. For $k = 1$, the best quantization is to pick $\mathbb{E}_\nu[X]$ with probability 1, incurring a loss of $I_\phi(X)$. For intermediate values of k, the solution is less trivial.
The following theorem shows[6] that the expected Bregman Information of the partitions (see Equation 5.14) and the loss in Bregman Information (see Equation 5.15) are equal.

Theorem 5.1 *Let X be a random variable that takes values in $\mathcal{X} = \{\vec{x}_i\}_{i=1}^n \subset \mathcal{S} \subseteq \mathbb{R}^d$ following the positive probability measure ν. Let $\{\mathcal{X}_h\}_{h=1}^k$ be a partitioning of $\mathcal{X}$ and let $\pi_h = \sum_{\vec{x}_i \in \mathcal{X}_h} \nu_i$ be the induced measure π on the partitions. Let X_h be the random variable that takes values in $\mathcal{X}_h$ following the probability measure ν_i/π_h for $\vec{x}_i \in \mathcal{X}_h$, for $h = 1, 2, \cdots, k$. Let $\mathcal{M} = \{\vec{\mu}_h\}_{h=1}^k$ with $\vec{\mu}_h \in ri(\mathcal{S})$ denote the set of the representatives of $\{X_h\}_{h=1}^k$, and M be a random variable that takes values in $\mathcal{M}$ following the probability distribution π. Then,*

$$L_\phi(M) = I_\phi(X) - I_\phi(M) = \mathbb{E}_\pi[I_\phi(X_h)] = \sum_{h=1}^{k} \pi_h \sum_{\vec{x}_i \in \mathcal{X}_h} \frac{\nu_i}{\pi_h} d_\phi(\vec{x}_i, \vec{\mu}_h) \tag{5.16}$$

Therefore, we can define the *Bregman Hard Clustering* problem as that of finding a partitioning of X, or equivalently, finding the random variable M, such that the *loss in Bregman Information* due to the quantization, $L_\phi(M) = I_\phi(X) - I_\phi(M)$, is *minimized*. Typically, clustering algorithms assume a uniform measure over the data, i.e. $\forall i = 1, 2, \cdots, n, \nu_i = 1/n$, which is clearly a special case of the general formulation above.
Note that $I_\phi(X)$ can be interpreted as the total Bregman Information, and $I_\phi(M)$ can be interpreted as the inter-cluster Bregman Information since it is a measure

[6] You can find the proof in Banerjee et al. (2005c).

Algorithm 2 Bregman hard clustering algorithm

procedure BREGMANHC($\mathcal{X}, \nu, d_\phi, k$)
  $\{\vec{\mu}_h\}_{h=1}^k \leftarrow$ selectRepresentatives(k, ri($\mathcal{S}$)) ▷ randomly, or other policies
  repeat
    ▷ assignment step
    for $h \leftarrow 1, k$ **do**
      $\mathcal{X}_h \leftarrow \emptyset$
    end for
    for $i \leftarrow 1, n$ **do**
      $h = \arg\min_{h'} d_\phi(\vec{x}_i, \vec{\mu}_{h'})$
      $\mathcal{X}_h \leftarrow \mathcal{X}_h \cup \{\vec{x}_i\}$
    end for
    ▷ re-estimation step
    for $h \leftarrow 1, k$ **do**
      $\pi_h \sum_{\vec{x}_i \in \mathcal{X}_h} \nu_i$
      $\vec{\mu}_h \leftarrow \frac{1}{\pi_h} \sum_{\vec{x}_i \in \mathcal{X}_h} \nu_i \vec{x}_i$
    end for
  until convergence
  $M^* \leftarrow \{\vec{\mu}_h\}_{h=1}^k$
  return M^*
end procedure

of divergence between cluster representatives, while $L_\phi(M)$ can be interpreted as the intra-cluster Bregman Information. Thus, Theorem 5.1 states that the total Bregman Information equals the sum of the intra-cluster Bregman Information and inter-cluster Bregman Information. This generalizes a result achieved for Euclidean distances (Duda et al., 2000).

Using Theorem 5.1, we can directly define the Bregman Hard Clustering problem.

Definition 5.3 (Bregman Hard Clustering) *Let X be a random variable that takes values in $\mathcal{X} = \{\vec{x}_i\}_{i=1}^n \subset \mathcal{S} \subseteq \mathbb{R}^d$ following the positive probability measure ν. Let $\{\mathcal{X}_h\}_{h=1}^k$ be a partitioning of $\mathcal{X}$ and let $\pi_h = \sum_{\vec{x}_i \in \mathcal{X}_h} \nu_i$ be the induced measure π on the partitions. Let X_h be the random variable that takes values in $\mathcal{X}_h$ following the probability measure ν_i/π_h for $\vec{x}_i \in \mathcal{X}_h$, for $h = 1, 2, \cdots, k$. Let $\mathcal{M} = \{\vec{\mu}_h\}_{h=1}^k$ with $\vec{\mu}_h \in ri(\mathcal{S})$ denote the set of the representatives of $\{X_h\}_{h=1}^k$, and M be a random variable that takes values in $\mathcal{M}$ following the probability distribution π.*

We define the **Bregman Hard Clustering problem** *as the problem of minimizing the loss in Bregman Information*

$$\min_M L_\phi(M) = I_\phi(X) - I_\phi(M) = \min_M \left(\sum_{h=1}^k \sum_{\vec{x}_i \in \mathcal{X}_h} \nu_i d_\phi(\vec{x}_i, \vec{\mu}_h) \right) \tag{5.17}$$

5.2.1 Bregman Hard Clustering algorithm

The objective function in Equation 5.17 suggests a natural iterative relocation algorithm for solving the Bregman hard clustering problem and it is easy to observe that the classical K-means and the Linde-Buzo-Gray (LBG) algorithm are special cases of the Bregman hard clustering problem for squared Euclidean distance and Itakura-Saito distance respectively.
Given a number k of the requested clusters, we initialize the k representatives with some policy, e.g. randomly like the K-means does with centroids. Then we have an *assignment step* that builds a partitioning, followed by a *re-estimation step* that recalculates the means of clusters. The two steps repeat until the convergence is achieved. The pseudo-code for the Bregman hard clustering algorithm is available in Algorithm 2.
The following propositions prove the convergence of the Bregman hard clustering algorithm.

Proposition 5.2 *The Bregman hard clustering algorithm monotonically decreases the loss function in Equation 5.17.*

Proposition 5.3 *The Bregman hard clustering algorithm terminates in a finite number of steps at a partition that is locally optimal, i.e. the total loss cannot be decreased by either the assignment step or by the re-estimation step (recalculation of the clusters means).*

Proofs are available in Banerjee et al. (2005c).
Like in the K-means, the Bregman hard clustering algorithm find a locally optimal solution which depends on the initialization of the cluster representatives.
In addition to local optimality, the Bregman hard clustering algorithm has the following interesting properties

- *Exhaustiveness*. The Bregman hard clustering algorithm with cluster centroids as optimal representatives works for all Bregman divergences and only for them, since the arithmetic mean is the best predictor only for Bregman divergences (Banerjee et al., 2005a, sec. 3).[7]
- *Linear separators*. For all Bregman divergences, the partitions induced by the Bregman hard clustering algorithm are separated by hyperplanes.
- *Scalability*. The computational complexity of each iteration of the Bregman hard clustering algorithm is linear in the number of data points and the number of desired clusters for all Bregman divergences.
- *Applicability to mixed data types*. The Bregman hard clustering algorithm is applicable to mixed data types that are commonly encountered in machine learning. One can choose different functions that are meaningful for different subsets of the features. The Bregman divergence corresponding to a

[7] It is possible to have similar alternate minimization based clustering algorithm for distance function that are not Bregman divergences. In this case the primary difference is that the optimal representative, where it exists, will no longer be the arithmetic mean or the expectation. Moreover, it is unique. Examples of such algorithms are the *convex K-means* (Modha and Spangler, 2003) and the LBG generalization (Linde et al., 1980).

convex combination of the component convex function can be then used to cluster the data.

5.3 Introduction to co-clustering problem

Although the co-clustering was introduced in the early 70s by Hartigan (1972), it has been neglected for many years. The clustering research has spent much effort on classic one-way data clustering. More recently, the one-way clustering has been used also for feature clustering. The latter serves both for dimensionality reduction (see subsubsection 3.6.1.3) and for collecting specific statistical informations. For instance, in text mining the feature clustering has been used to create class models of word occurrence (Pereira et al., 1993) and to reduce the dimensionality both in classification problems (Baker and McCallum, 1998; Slonim and Tishby, 2001) and in clustering problems (Slonim and Tishby, 2000).
Recently, the research effort in the co-clustering direction is increased, for several reasons. One motivation is the need of contextually performing the dimensionality reduction during the data clustering, instead of performing a generic dimensionality reduction as a preprocessing step for data clustering. Another statistically important reason is the necessity of revealing the interplay between similar data objects and their feature clusters. This is a statistical information we can obtain only by solving a co-clustering problem.
There are several application domains interested in this type of clustering, such as text mining (Dhillon and Guan, 2003a; Dhillon, 2001; Dhillon and Guan, 2003b; Dhillon et al., 2003b), bioinformatics (Cheng and Church, 2000; Cho et al., 2004a; Tchagang and Tewfik, 2005; Yang et al., 2003), images analysis (Guan et al., 2005; Qiu, 2004), Natural Language Processing (NLP) (Freitag, 2004; Rohwer and Freitag, 2004).
The main goal of the co-clustering problem is to find co-clusters instead of simple clusters. A co-cluster is a group of objects and features that are inter-related. The features in a co-cluster are the attributes used for choosing the objects to put in that co-cluster. For example, in the analysis of DNA microarrays this means to find submatrices composed of subgroups of genes and subgroups of conditions, where the genes of a submatrix exhibit highly correlated activities for every condition in the same submatrix.
There are different approaches to perform the co-clustering. Recent approaches employ a spectral graph framework (Dhillon, 2001), a framework based on the information theory (Dhillon and Guan, 2003a,b; Dhillon et al., 2003b), a minimum sum-squared residue framework (Cho et al., 2004a), a *Block Value Decomposition (BVD)* framework (Long et al., 2005), and many others (Cheng and Church, 2000; Tchagang and Tewfik, 2005; Yang et al., 2003). Some of them have the great limitation of finding just one co-cluster for each algorithm execution, such as the solutions proposed in Cheng and Church (2000) or in Yang et al. (2003).
The most interesting works are the aforementioned information-theoretic framework and the minimum sum-squared residue framework. Both use a generalized K-means like strategy, i.e. an alternate minimization scheme for cluster-

ing both dimensions of a data matrix. In fact, such works are the main motivations that have led the researchers to the creation of a more general framework based on Minimum Bregman Information (MBI) principle (Banerjee et al., 2004a,b, 2007; Deodhar and Ghosh, 2007). Such framework was created for a hard co-clustering problem, but extensions to the soft co-clustering problem are also available (Gupta and Chandola, 2006; Shafiei and Milios, 2006). Such a general framework allows not to make assumptions on the data matrices, like the non-negativeness if we treat them as contingency tables. In fact, the fundamental properties (e.g. the convergence of the algorithm which solves the problem) hold in the general case.
Furthermore, this framework allows performing other interesting operations on data matrices, such as the *missing values prediction* and the *matrix approximation (or compression)*. These peculiarities make this framework a more general unsupervised tool for data analysis.
In the next sections we provide an exhaustive presentation of the *Minimum Bregman Information (MBI) principle for hard co-clustering* and briefly review its ability of predicting missing values and approximating matrices.

5.4 The Bregman hard co-clustering framework

The application of the MBI principle to the co-clustering problem leads to a general framework. To appreciate the generalization, it is helpful to view the hard co-clustering as a lossy data compression problem where, given a specified number of row and column clusters, one attempts to retain as much information as possible about the original data matrix in terms of statistics based on the co-clustering.[8] The main idea is that the reconstruction based on the co-clustering should result in the same set of user-specified statistics as the original matrix. There are two key components in formulating a co-clustering problem: (i) choosing a set of critical co-clustering-based statistics of the original data matrix that need to be preserved, and (ii) selecting an appropriate measure to quantify the information loss or discrepancy between the original data matrix and the compressed representation provided by the co-clustering.
According to these two key components, we can obtain a specific instance of the co-clustering framework, either already formulated in the past or fresh ones. For example, in the information-theoretic framework (Dhillon et al., 2003b) the summary statistics preserved are the row and column averages and the co-cluster average; further, the quality of the compressed representation is measured in terms of the sum of element-wise I-divergence. In contrast, in the work of Cheng and Church (2000) the row and column averages of each co-cluster are preserved and the discrepancy between the original data matrix and its approximation is measured in terms of the sum of element-wise squared deviation. Surprisingly, we will show in the sequel that also a one-way clustering algorithm such as the classical K-means can be derived from this framework.

[8] This viewpoint was first employed in the information-theoretic framework by Dhillon et al. (2003b).

5.4.1 Data matrices and random variables

The problem focuses on the co-clustering of a specified $m \times n$ data matrix $\mathbf{Z}$. Let each entry of $\mathbf{Z} = [z_{uv}]$ takes values in the convex set $\mathcal{S} = \text{dom}(\phi)$, for example,[9] $\mathcal{S} = \mathbb{R}$ for $\phi(z) = z^2$ or $\mathcal{S} = \mathbb{R}_+$ for $\phi(z) = z \log z - z$. Hence, $\mathbf{Z} \in \mathcal{S}^{m \times n}$. Observe that this implies the employment of a much larger class of matrices than those used in Cho et al. (2004a) and in Dhillon et al. (2003b).
Given the data matrix $\mathbf{Z}$, we consider a random variable Z that takes values in $\mathbf{Z}$ following a probability distribution as described below. Let U be a random variable that takes values in the set of row indices $\{1, 2, \cdots, m\}$, and let V be a random variable that takes values in the set of the column indices $\{1, 2, \cdots, n\}$. Let (U, V) be distributed according to a probability measure $w = \{w_{uv} \colon u = 1, 2, \cdots, m\, v = 1, 2, \cdots, n\}$, which is either pre-specified or set to be the uniform distribution.[10] Let Z be a (U,V)-measurable random variable that takes values in $\mathbf{Z}$ following w, that is $p(Z(u, v) = z_{uv}) = w_{uv}$. Clearly, for a given matrix $\mathbf{Z}$, the random variable Z is a deterministic function of the random variable (U, V).

5.4.2 General problem statement

A $k \times l$ hard co-clustering is a pair of functions

$$\rho : \{1, 2, \cdots, m\} \to \{1, 2, \cdots, k\}$$
$$\gamma : \{1, 2, \cdots, n\} \to \{1, 2, \cdots, l\}$$

where k and l are the number of row clusters and column clusters requested respectively.
Let $\hat{U}$ and $\hat{V}$ be random variables that takes values in $\{1, 2, \cdots, k\}$ and $\{1, 2, \cdots, l\}$ such that $\hat{U} = \rho(U)$ and $\hat{V} = \gamma(V)$. Let $\mathbf{Z} = [\hat{z}_{uv}] \in \mathcal{S}^{m \times n}$ be an approximation for the data matrix $\mathbf{Z}$ such that $\hat{\mathbf{Z}}$ only depends on a given co-clustering (ρ, γ) and certain summary statistics derived from the co-clustering. Let $\hat{Z}$ be a (U, V)-measurable random variable that takes values in this approximate matrix $\hat{\mathbf{Z}}$ following the probability measure w, that is, $p(\hat{Z}(U, V) = \hat{z}_{uv}) = w_{uv}$. Then the goodness of the underlying co-clustering can be measured in terms of the expected distortion between Z and $\hat{Z}$

$$\mathbb{E}[d_\phi(Z, \hat{Z})] \tag{5.18}$$

which results in the distortion between the original matrix $\mathbf{Z}$ and the approximation $\hat{\mathbf{Z}}$. The better the approximation, the higher the quality of the co-clustering.

[9] $\mathcal{S}$ needs not necessarily be a subset of $\mathbb{R}$, but it is convenient to assume so for ease of exposition. In general, the elements of the matrix $\mathbf{Z}$ can take values over any convex domain with a well-defined Bregman divergence.

[10] Associating a measure with the elements of a matrix is not common, but this allows us dealing with a wider variety of situations including the modeling of matrices with missing values. Further, several quantities of interest, such as row/column/block averages, can now be described in terms of conditional expectations. From now on, all expectations are implicitly intended with respect to the measure w, unless differently specified.

5.4.3 Co-clustering bases

Various possible co-clustering schemes arise due to the preservation of different choices of summary statistics. A co-clustering basis corresponds to each scheme; such a basis is defined in terms of conditional expectation-based statistics to be preserved.

Let us fix a co-clustering (ρ, γ). Given the co-clustering, there are essentially four random variables of interest: $U, V, \hat{U}$ and $\hat{V}$. To these, we add two random variables $U_\emptyset$ and $V_\emptyset$ corresponding to the constant random variables over the rows and columns respectively, to make enumeration easier. Let $\Gamma_1 = \{U_\emptyset, V_\emptyset, \hat{U}, \hat{V}, U, V\}$ be the set of the above random variables. The goal is to approximate the random variable Z using (possibly multiple) conditional expectations of Z where the conditioning is done on one or more of the random variables in Γ_1. Observe that choosing one or more random variables to condition on is equivalent to choosing a sub-σ-algebra[11] $\mathcal{G}$ of Z. Z is approximated using conditional expectations $\mathbb{E}[Z|\mathcal{G}]$ since such expectations are the optimal approximations of the true Z variables with respect to any Bregman divergence among all $\mathcal{G}$-measurable functions. Since duplication of information in the preserved conditional expectations does not yield different approximations, it is obvious to consider only that combinations of random variables in Γ_1 that will lead to a unique set of summary statistics. First, we can note that some of random variables in Γ_1 are measurable with respect to some others. In other words, some variables are just "high resolution" versions of some others so that conditioning on certain sets of members of Γ_1 is equivalent to conditioning on the subset with respect to which the rest are measurable. For example, $\mathbb{E}[Z|U, \hat{U}, V_\emptyset, \hat{V}] = \mathbb{E}[Z|U, \hat{V}]$, since $\hat{U}$ is U-measurable, and $V_\emptyset$ is $\hat{V}$-measurable. In fact, due to the natural ordering of the random variables $\{U_\emptyset, \hat{U}, U\}$ and $\{V_\emptyset, \hat{V}, V\}$ in terms of measurability, only the row and column random variables of the highest granularity matter. Hence, there are only nine unique sub-σ-algebras of Z based on which conditional expectations may be taken. Let us denote this by

$$\Gamma_2 = \{\{U_\emptyset, V_\emptyset\}, \{U_\emptyset, \hat{V}\}, \{U_\emptyset, V\}, \{\hat{U}, V_\emptyset\}, \{\hat{U}, \hat{V}\}, \{\hat{U}, V\}, \{U, V_\emptyset\}, \{U, \hat{V}\}, \{U, V\}\}$$

The set Γ_2 determines the set of all summary statistics that one maybe interested in preserving. A particular choice of an element of Γ_2 leads to an approximation scheme where the reconstruction matrix preserves the corresponding summary statistics.

Anyway, one may want to preserve possibly more than one summary statistics. In fact, one could consider all possible subsets of Γ_2. Of these, some combinations of summary statistics are effectively equivalent, for example, $\{\{\hat{U}, V_\emptyset\}, \{U_\emptyset, \hat{V}\}, \{\hat{U}, \hat{V}\}\}$ and $\{\{\hat{U}, \hat{V}\}\}$, whereas some others are trivial and even independent of the co-clustering, for example, $\{\{U_\emptyset, V_\emptyset\}\}$ and $\{\{U_\emptyset, V\}, \{U, V_\emptyset\}\}$. The Bregman

[11] A σ-algebra is a collection of sets that includes the empty-set and is closed with respect to complements, countable unions and intersections. Further, $\mathcal{G}_1$ is a sub-σ-algebra of a σ-algebra $\mathcal{G}$ (or a $\mathcal{G}$-measurable random variable) if $\mathcal{G}_1$ is itself a σ-algebra and $\mathcal{G}_1 \subseteq \mathcal{G}$.

framework for the hard co-clustering only focuses on unique and nontrivial combinations of the elements of Γ_2, called *co-clustering bases*. The definition of the co-clustering basis follows.

Definition 5.4 (Co-clustering basis) *A* **co-clustering basis** $\mathcal{C}$ *is a set of elements of* Γ_2,[12] *i.e. an element of the power set* 2^{Γ_2}*, which satisfies the following two conditions*

1. *There exist* $\mathcal{G}_1, \mathcal{G}_2 \in C$ *such that* $\hat{U} \in \mathcal{G}_1$ *and* $\hat{V} \in \mathcal{G}_2$
2. *There do not exist* $\mathcal{G}_1, \mathcal{G}_2 \in C$, $\mathcal{G}_1 \neq \mathcal{G}_2$ *such that* $\mathcal{G}_2$ *is a sub-*σ*-algebra of* $\mathcal{G}_1$

The first condition ensures that the approximation depends on the co-clustering while the second condition ensures that for any pair of $\mathcal{G}_1, \mathcal{G}_2$, the conditional expectation $\mathbb{E}[Z|\mathcal{G}_2]$ cannot be obtained from $\mathbb{E}[Z|\mathcal{G}_1]$. The latter ensures that the approximation obtained using the basis $\mathcal{C}$ is not identical to that obtained using $\mathcal{C} \setminus \mathcal{G}_2$.
The following theorem shows that there are only six possible co-clustering bases, each of which leads to a distinct matrix approximation scheme.

Theorem 5.2 *Given the random variable* Z*, there are only six distinct co-clustering bases that approximate* Z *using conditional expectations of* Z *given combinations of the row and column random variables* $\{U, V, \hat{U}, \hat{V}\}$*. The six bases correspond to the sets*

$$\mathcal{C}_1 = \{\{\hat{U}\}, \{\hat{V}\}\}, \mathcal{C}_2 = \{\{\hat{U}, \hat{V}\}\},$$
$$\mathcal{C}_3 = \{\{\hat{U}, \hat{V}\}, \{U\}\}, \mathcal{C}_4 = \{\{\hat{U}, \hat{V}\}, \{V\}\},$$
$$\mathcal{C}_5 = \{\{\hat{U}, \hat{V}\}, \{U\}, \{V\}\}, \mathcal{C}_3 = \{\{U, \hat{V}\}, \{\hat{U}, V\}\}.$$

The proof is omitted and is available in Banerjee et al. (2007).
The sets $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_5$ and $\mathcal{C}_6$ are symmetric in the row and column random variables whereas the remaining two bases are not. Further, if we have access to $\{\mathbb{E}[Z|\mathcal{G}] \colon \mathcal{G} \in \mathcal{C}_i\}$, for some $i = 1, 2, \cdots, 6$, then we can compute $\{\mathbb{E}[Z|\mathcal{G}] \colon \mathcal{G} \in \mathcal{C}_j\}$ for all $j = 1, \cdots, i$, $i \neq 4$, $j \neq 3$. Loosely speaking, this means that the constraint set $\mathcal{C}_i$ is more complex than $\mathcal{C}_j$ for all $j = i, \cdots, 6$, $i \neq 4$, $j \neq 3$. From a practical perspective, the more complex the set of constraints, the more information we retain about Z.
The abstraction of the Bregman framework allows to handle all schemes in a systematic way. Let us consider a co-clustering basis $\mathcal{C} \in \{\mathcal{C}_i\}_{i=1}^6$ as the pertinent one. Given the choice of a particular basis, it is needed to decide on the best reconstruction $\hat{Z}$ for a given co-clustering (ρ, γ). Then the general co-clustering problem will effectively reduce to one of finding an optimal co-clustering (ρ^*, γ^*) whose reconstruction has the lowest approximation error with respect to the original data matrix Z.[13]

[12] Note that each element of Γ_2 corresponds to a unique sub-σ-algebra of Z and hence we use identical notation for elements of the co-clustering bases and the corresponding sub-σ-algebras.

[13] It is interesting to note that the role played by the approximated matrix reflects the role of the Bregman representative in one-way Bregman hard clustering problem (see subsection 5.1.3).

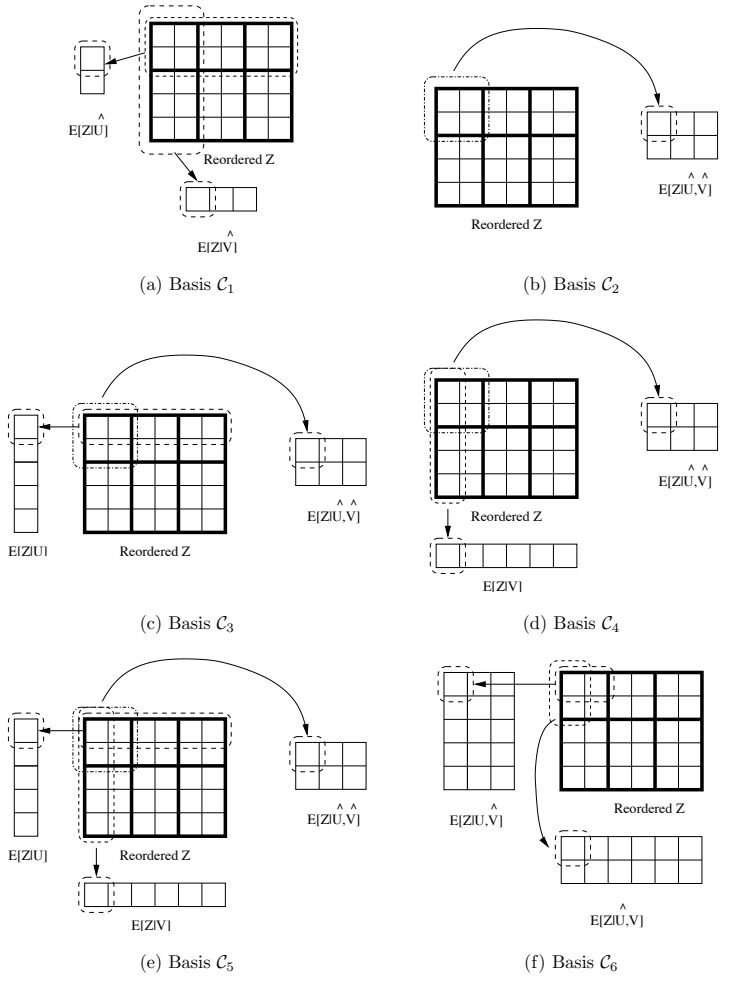


Figure 5.1: *Schematic diagram of the six co-clustering bases. In each case, the summary statistics used for reconstruction are the expectations taken over the corresponding dotted regions. The matrices are "reordered", i.e. the rows and the columns are arranged according to cluster order to reveal the various co-clusters. (Banerjee et al., 2007, fig. 1)*

5.4.4 Minimum Bregman Information

For a given co-clustering (ρ, γ) and a given co-clustering basis C, the best approximation $\hat{Z}$ of Z is obtained by means of the *Minimum Bregman Information (MBI) principle*. For a general co-clustering basis C, we search for the best approximation $\hat{Z}$ among all matrices Z' that preserve all the summary statistics relevant to C. Let $\mathcal{S}_A$ denote a class of random variables such that every Z' in the class satisfies the following *linear constraints*, that is

$$\mathcal{S}_A = \{Z' | \mathbb{E}[Z|\mathcal{G}] = \mathbb{E}[Z'|\mathcal{G}], \forall \mathcal{G} \in \mathcal{C}\} \tag{5.19}$$

Now, let us select the random variable $\hat{Z}_A \in \mathcal{S}_A$ with the minimum Bregman information as the best approximation

$$\hat{Z}_A \equiv \arg \min_{Z' \in \mathcal{S}_A} I_\phi(Z') \tag{5.20}$$

The basic philosophy behind the MBI principle is that the best approximation given certain information is one that does not make any extra assumptions over the available information. Mathematically, the notion of "no extra assumptions" or maximal uncertainty translates to Minimum Bregman Information (MBI) while the available information is provided by the linear constraints that preserve the specified statistics.
As the following examples show, the Minimum Bregman Information (MBI) principle generalizes famous principles, such as the *maximum entropy* (Cover and Thomas, 1991) and the *standard least squares* (Csiszár, 1991).

Example 5.5 (Standard least squares principle) *From Example 5.3, we observe that the Bregman information of a random variable Z following a uniform distribution over the elements of a matrix $\mathbf{Z}$ is given by $\frac{1}{mn}\|\mathbf{Z}\|_F^2$ up to an additive constant. Hence, minimizing the Bregman information in this case is equivalent to minimizing the Frobenius norm*[14] *of the matrix, which in turn implies that the* standard least squares principle *is a special case of the* MBI principle *corresponding to the squared Euclidean distance.*

Example 5.6 (Maximum entropy principle) *From Example 5.4, we observe that the Bregman information of a random variable Z following a uniform distribution over the joint probability values of two other random variables X e Y is given by $-\frac{1}{mn}H(p(X,Y))$ up to an additive constant, i.e. it is negatively related to entropy $H(\cdot)$ of the joint distribution of X and Y. Hence, minimizing the Bregman information in equivalent to maximizing the entropy demonstrating that the* maximum entropy principle *is a special case of the* MBI principle *corresponding to the I-divergence (which in turns is a generalization of the KL-divergence).*

Note that for characterizing the solution, the Lagrangian dual problem is considered. Let $J(Z', \Lambda)$ be the Lagrangian of the MBI problem in Equation 5.20

$$\begin{aligned} J(Z', \Lambda) &= I_\phi(Z') + \sum_{r=1}^{s} \mathbb{E}_{\mathcal{G}_r}\left[\frac{\Lambda_{\mathcal{G}_r}}{w_{\mathcal{G}_r}}(\mathbb{E}[Z'|\mathcal{G}_r] - \mathbb{E}[Z|\mathcal{G}_r])\right] = \\ &= \mathbb{E}[\phi(Z')] - \phi(\mathbb{E}[Z']) + \sum_{r=1}^{s} \mathbb{E}_{\mathcal{G}_r}\left[\frac{\Lambda_{\mathcal{G}_r}}{w_{\mathcal{G}_r}}(\mathbb{E}[Z'|\mathcal{G}_r] - \mathbb{E}[Z|\mathcal{G}_r])\right] \end{aligned} \tag{5.21}$$

where $\Lambda_{\mathcal{G}_r}$ is a deterministic function of the random variable $\mathcal{G}_r$ and equals the appropriate Lagrange multipliers when $\mathcal{G}_r$ is specified. The Lagrange dual problem is

$$L(\Lambda) = \inf_{Z'} J(Z', \Lambda) \tag{5.22}$$

and it is concave in Λ.
The solution for the Lagrangian dual form of the MBI problem is characterized by the following theorem.

[14] The Frobenius norm is the Euclidean L_2 norm extended to the matrices.

Basis	Lagrange multipliers	Approximation $\hat{Z}_A$
$\mathcal{C}_1$	$\Lambda^*_{\hat{U}} = -2w_{\hat{U}}(\mathbb{E}[Z\|\hat{U}] - \mathbb{E}[Z])$, $\Lambda^*_{\hat{V}} = -2w_{\hat{U}}(\mathbb{E}[Z\|\hat{V}] - \mathbb{E}[Z])$	$\mathbb{E}[Z\|\hat{U}] + \mathbb{E}[Z\|\hat{V}] - \mathbb{E}[Z]$
$\mathcal{C}_2$	$\Lambda^*_{\hat{U},\hat{V}} = -2w_{\hat{U},\hat{V}}(\mathbb{E}[Z\|\hat{U},\hat{V}] - \mathbb{E}[Z])$	$\mathbb{E}[Z\|\hat{U},\hat{V}]$
$\mathcal{C}_3$	$\Lambda^*_{\hat{U},\hat{V}} = -2w_{\hat{U},\hat{V}}(\mathbb{E}[Z\|\hat{U},\hat{V}] - \mathbb{E}[Z])$, $\Lambda^*_U = -2w_U(\mathbb{E}[Z\|U] - \mathbb{E}[Z\|\hat{U}])$	$\mathbb{E}[Z\|\hat{U},\hat{V}] + \mathbb{E}[Z\|U] + \mathbb{E}[Z\|\hat{U}]$
$\mathcal{C}_4$	$\Lambda^*_{\hat{U},\hat{V}} = -2w_{\hat{U},\hat{V}}(\mathbb{E}[Z\|\hat{U},\hat{V}] - \mathbb{E}[Z])$, $\Lambda^*_V = -2w_V(\mathbb{E}[Z\|V] - \mathbb{E}[Z\|\hat{V}])$	$\mathbb{E}[Z\|\hat{U},\hat{V}] + \mathbb{E}[Z\|V] + \mathbb{E}[Z\|\hat{V}]$
$\mathcal{C}_5$	$\Lambda^*_{\hat{U},\hat{V}} = -2w_{\hat{U},\hat{V}}(\mathbb{E}[Z\|\hat{U},\hat{V}] - \mathbb{E}[Z])$, $\Lambda^*_U = -2w_U(\mathbb{E}[Z\|U] - \mathbb{E}[Z\|\hat{U}])$, $\Lambda^*_V = -2w_V(\mathbb{E}[Z\|V] - \mathbb{E}[Z\|\hat{V}])$	$\mathbb{E}[Z\|\hat{U},\hat{V}]+\mathbb{E}[Z\|U]+\mathbb{E}[Z\|\hat{U}]+\mathbb{E}[Z\|V]+\mathbb{E}[Z\|\hat{V}]$
$\mathcal{C}_6$	$\Lambda^*_{\hat{U},V} = -2w_{\hat{U},V}(\mathbb{E}[Z\|\hat{U},V] - \frac{\mathbb{E}[Z]}{2} - \frac{\mathbb{E}[Z\|\hat{U},\hat{V}]}{2})$, $\Lambda^*_{U,\hat{V}} = -2w_{U,\hat{V}}(\mathbb{E}[Z\|U,\hat{V}] - \frac{\mathbb{E}[Z]}{2} - \frac{\mathbb{E}[Z\|\hat{U},\hat{V}]}{2})$	$\mathbb{E}[Z\|U,\hat{V}] + \mathbb{E}[Z\|\hat{U},V] - \mathbb{E}[Z\|\hat{U},\hat{V}]$

Table 5.2: *MBI solution and optimal Lagrange multipliers for Squared Euclidean distance.*

Theorem 5.3 (MBI solution) *For any random variable Z and a specified co-clustering basis $\mathcal{C} = \{\mathcal{G}_r\}_{r=1}^s$, the solution $\hat{Z}_A$ to the problem in Equation 5.21 (and therefore to the original problem in Equation 5.20) in given by*

$$\nabla\phi(\hat{Z}_A) = \nabla\phi(\mathbb{E}[Z]) - \sum_{r=1}^{s} \frac{\Lambda^*_{\mathcal{G}_r}}{w_{\mathcal{G}_r}} \tag{5.23}$$

where $w_{\mathcal{G}_r}$ is the measure corresponding to $\mathcal{G}_r$ and $\{\Lambda^*_{\mathcal{G}_r}\}_{r=1}^s$ are the optimal Lagrange multipliers corresponding to the set of linear constraints

$$\forall r = 1, 2, \cdots, s, \mathbb{E}[Z^{'}|\mathcal{G}_r] = \mathbb{E}[Z|\mathcal{G}_r] \tag{5.24}$$

A proof is available in Banerjee et al. (2007).
Note that the strict convexity of ϕ ensures that $\nabla\phi$ is a bijection so that Equation 5.23 uniquely determines the approximation $\hat{Z}_A$.
For the squared Euclidean distance and for I-divergence we have the MBI solutions and optimal Lagrange multipliers in Table 5.2 and Table 5.3 respectively. However, let us highlight that the Lagrange dual $L(\Lambda)$ of Bregman information is concave in Λ for all bases, but strictly concave only for the basis $\mathcal{C}_2$. So, the multipliers shown in Table 5.2 and Table 5.3 are only one of the possible maximizers of $L(\Lambda)$ (expect for $\mathcal{C}_2$).
The matrix approximation $\hat{\mathbf{Z}}_A$ corresponding to the MBI solution $\hat{Z}_A$ can be obtained by instantiating $\hat{Z}_A$ for specific choices of U and V.
Finally, it can be shown that the MBI solution is optimal, i.e. it minimizes the approximation error with respect to the original matrix among all reconstructions that correspond to measurable functions of available summary statistics (Banerjee et al., 2007, sec. 4.4).

Basis	Lagrange multipliers	Approximation $\hat{Z}_A$
$\mathcal{C}_1$	$\Lambda^*_{\hat{U}} = -2w_{\hat{U}} \log\left(\frac{\mathbb{E}[Z\|\hat{U}]}{\mathbb{E}[Z])}\right)$, $\Lambda^*_{\hat{V}} = -2w_{\hat{U}} \log\left(\frac{\mathbb{E}[Z\|\hat{V}]}{\mathbb{E}[Z]}\right)$	$\frac{\mathbb{E}[Z\|\hat{U}]\times\mathbb{E}[Z\|\hat{V}]}{\mathbb{E}[Z]}$
$\mathcal{C}_2$	$\Lambda^*_{\hat{U},\hat{V}} = -2w_{\hat{U},\hat{V}} \log\left(\frac{\mathbb{E}[Z\|\hat{U},\hat{V}]}{\mathbb{E}[Z]}\right)$	$\mathbb{E}[Z\|\hat{U},\hat{V}]$
$\mathcal{C}_3$	$\Lambda^*_{\hat{U},\hat{V}} = -2w_{\hat{U},\hat{V}} \log\left(\frac{\mathbb{E}[Z\|\hat{U},\hat{V}]}{\mathbb{E}[Z]}\right)$, $\Lambda^*_{U} = -2w_{U} \log\left(\frac{\mathbb{E}[Z\|U]}{\mathbb{E}[Z\|\hat{U}]}\right)$	$\frac{\mathbb{E}[Z\|\hat{U},\hat{V}]\times\mathbb{E}[Z\|U]}{\mathbb{E}[Z\|\hat{U}]}$
$\mathcal{C}_4$	$\Lambda^*_{\hat{U},\hat{V}} = -2w_{\hat{U},\hat{V}} \log\left(\frac{\mathbb{E}[Z\|\hat{U},\hat{V}]}{\mathbb{E}[Z]}\right)$, $\Lambda^*_{V} = -2w_{V} \log\left(\frac{\mathbb{E}[Z\|V]}{\mathbb{E}[Z\|\hat{V}]}\right)$	$\frac{\mathbb{E}[Z\|\hat{U},\hat{V}]\times\mathbb{E}[Z\|V]}{\mathbb{E}[Z\|\hat{V}]}$
$\mathcal{C}_5$	$\Lambda^*_{\hat{U},\hat{V}} = -2w_{\hat{U},\hat{V}} \log\left(\frac{\mathbb{E}[Z\|\hat{U},\hat{V}]}{\mathbb{E}[Z]}\right)$, $\Lambda^*_{U} = -2w_{U} \log\left(\frac{\mathbb{E}[Z\|U]}{\mathbb{E}[Z\|\hat{U}]}\right)$, $\Lambda^*_{V} = -2w_{V} \log\left(\frac{\mathbb{E}[Z\|V]}{\mathbb{E}[Z\|\hat{V}]}\right)$	$\frac{\mathbb{E}[Z\|\hat{U},\hat{V}]\times\mathbb{E}[Z\|U]\times\mathbb{E}[Z\|\hat{U}]}{\mathbb{E}[Z\|V]+\mathbb{E}[Z\|\hat{V}]}$
$\mathcal{C}_6$	$\Lambda^*_{\hat{U},V} = -2w_{\hat{U},V} \log\left(\frac{\mathbb{E}[Z\|\hat{U},V]}{(\mathbb{E}[Z]\mathbb{E}[Z\|\hat{U}\hat{V}])^{1/2}}\right)$, $\Lambda^*_{U,\hat{V}} = -2w_{U,\hat{V}} \log\left(\frac{\mathbb{E}[Z\|U,\hat{V}]}{(\mathbb{E}[Z]\mathbb{E}[Z\|\hat{U}\hat{V}])^{1/2}}\right)$	$\frac{\mathbb{E}[Z\|U,\hat{V}]\times\mathbb{E}[Z\|\hat{U},V]}{\mathbb{E}[Z\|\hat{U},\hat{V}]}$

Table 5.3: *MBI solution and optimal Lagrange multipliers for I-divergence.*

5.4.5 Hard co-clustering problem formulation

The expected Bregman divergence between the given random variable Z and the minimum Bregman information solution $\hat{Z}$ provides an elegant way to quantify the goodness of a co-clustering. This expected Bregman divergence is also exactly equal to the loss in Bregman information due to co-clustering as the following lemma shows. This equivalence provides another fashionable interpretation for the Bregman co-clustering formulation.

Lemma 5.4 *For any random variable Z,*

$$\mathbb{E}[d_\phi(Z, \hat{Z})] = I_\phi(Z) - I_\phi(\hat{Z}) \tag{5.25}$$

where $\hat{Z} = \hat{Z}_A$ defined in Equation 5.20.

The generalized co-clustering problem formulation follows.

Definition 5.5 (General Bregman Co-clustering problem) *Given k, l, a Bregman divergence d_ϕ, a random variable Z following a nonnegative measure w over the data matrix $\mathbf{Z} \in \mathcal{S}^{m\times n}$, and a co-clustering basis $\mathcal{C}$, we wish to find a co-clustering (ρ^*, γ^*) such that*

$$(\rho^*, \gamma^*) = \arg\min_{(\rho,\gamma)} \mathbb{E}[d_\phi(Z, \hat{Z})] = \arg\min_{(\rho,\gamma)} (I_\phi(Z) - I_\phi(\hat{Z})) = \arg\max_{(\rho,\gamma)} I_\phi(\hat{Z}) \tag{5.26}$$

where $\hat{Z} = \arg\min_{Z'\in\mathcal{S}_A} I_\phi(Z')$ is the MBI solution of the MBI problem as defined in Equation 5.20.

The general problem is NP-hard by a reduction from the K-means problem (Banerjee et al., 2007). Hence, it is difficult to obtain a globally optimal solution efficiently.

5.4.6 A meta algorithm

In this section we present the alternating minimization scheme for the general Bregman hard co-clustering problem (see Equation 5.26), which is a *meta algorithm* because it is a general scheme applicable for all co-clustering bases and all Bregman divergences. Therefore, the meta algorithm also applies to previously known problems, such as the information-theoretic co-clustering problem (Dhillon et al., 2003b) as well as to unknown specific co-clustering formulations. Throughout this section we assume that the underlying measure w, the Bregman divergence $d_\phi(\cdot)$, the data matrix $\mathbf{Z}$, the number of row clusters k, the number of column clusters l and the co-clustering basis $\mathcal{C}$ are specified and fixed.

5.4.6.1 The idea

The essence of the meta algorithm is summarized as follows

1. Start with an arbitrary row and column clustering, say (ρ^0, γ^0).[15] Set $t = 0$.
2. Repeat the following sub-steps till convergence
   (a) With respect to co-clustering (ρ^t, γ^t), compute the matrix approximation $\hat{Z}^t$, i.e. the MBI solution, by solving the MBI problem of Equation 5.20.
   (b) Hold the column clustering γ^t fixed and find a better row clustering, say ρ^{t+1}. Set $\gamma^{t+1} = \gamma^t$. Set $t = t + 1$.
   (c) Hold the row clustering ρ^t fixed and find a better column clustering, say γ^{t+1}. Set $\rho^{t+1} = \rho^t$. Set $t = t + 1$.

In the sequel we will state the theorem which ensures that such a meta algorithm converges to a local minimum in a finite number of steps.[16] As it is clear from the outline above, a key step of the algorithm is finding a solution of the MBI problem of Equation 5.20. Further, since the number of possible row (or column) clusterings is exponential in the number of rows (or columns), it is also essential to have an efficient means for determining the best row (or column) clustering for a fixed choice of the column (or row) clustering and the MBI solution.
Fortunately, the *separability property* explained in the next section allows independent optimal updates of the cluster assignments of every rows and/or columns.

[15] Note that this is the same initialization performed by the K-means and other similar algorithms, even the Bregman hard clustering algorithm in subsection 5.2.1. The only difference is that in this case we initialize a co-clustering, i.e. a row clustering and a column clustering, instead of a simple one-way clustering.

[16] In fact, any ordering of the step 2 sub-steps gives the same guarantees. Each of them can only improve the quality of the current approximation, therefore any ordering is sufficient to reach a local minimum.

5.4.6.2 The separability property

Let us consider the quality of a candidate row (or column) clustering ρ in step $2(b)$ (or γ in step $2(c)$) for a fixed choice of column (or row) clustering and MBI solution parameters. Since the goal is to obtain an accurate reconstruction of the original data matrix, a natural choice is to consider the expected Bregman distortion between the original Z and a reconstruction $\tilde{Z}$ based on the row (or column) clustering ρ (or γ) while keeping everything else fixed. To characterize this reconstruction, we employ the functional form for the MBI solution $\hat{Z}$ given in Theorem 5.3. In general, the formula in Theorem 5.3 yields a unique reconstruction $\tilde{Z}$ for any set of Lagrange multipliers Λ and (ρ, γ), since $\nabla\phi$ is monotonic, hence invertible (Banerjee et al., 2004c). To underscore the dependence of $\tilde{Z}$ on the Lagrange multipliers, we use the notation $\tilde{Z} = \zeta(\rho, \gamma, \Lambda) = (\nabla\phi)^{-1}(\nabla\phi(\mathbb{E}[Z]) - \sum_{r=1}^{s} \Lambda_{\mathcal{G}_r}/w_{\mathcal{G}_r})$. The quality of a candidate row (or column) clustering can now be quantified in terms of the accuracy of the corresponding $\tilde{Z}$ where the other two arguments, i.e. the column (or row) clustering and Lagrange multipliers, are fixed. In particular, $\hat{Z} = \zeta(\rho, \gamma, \Lambda^*)$ is the approximation corresponding to the optimal Lagrange multipliers Λ^*.

Given a set of Lagrange multipliers Λ, we now consider updating the current co-cluster assignments (ρ, γ) in order to improve the current approximation $\tilde{Z} = \zeta(\rho, \gamma, \Lambda)$. Although $\tilde{Z}$ looks complex, the fact that $\nabla\phi$ is invertible ensures that each element $\tilde{z}_{uv}$ in the matrix $\tilde{\mathbf{Z}}$ corresponding to $\tilde{Z}$ depends only on $(u, \rho(u), v, \gamma(v))$ for a given Λ. Hence, for any given Λ, there exists a function $\xi(\cdot)$ such that the point-wise distortion $d_\phi(z_{uv}, \tilde{z}_{uv})$ can be expressed as $\xi(u, \rho(u), v, \gamma(v))$, i.e. it depends only on the corresponding row/column and cluster assignments. Since the expected distortion $\mathbb{E}[d_\phi(Z, \tilde{Z})]$ is a weighted sum of the point-wise distortions, it satisfies an interesting *separability property* that allows the current row (or column) cluster assignments to be efficiently updated. In particular, for any given Λ, the expected distortion $\mathbb{E}[d_\phi(Z, \tilde{Z})]$ can be expressed as the sum of contributions from the rows (or columns) where each row (or column) contribution depends only on the row (or column) and its current cluster assignment. Note that this separability property is similar to that of the K-means objective function, which can also be expressed as the sum of terms corresponding to each point and its cluster assignment. The separability property allows independent updates of the cluster assignments of every row (or column). Further, for a fixed Λ and γ, since the total approximation error is the sum over the approximation errors due to each row (or column) and its cluster assignment, greedy cluster assignments of the individual rows result in a globally optimal row clustering ρ for the given Λ and γ. An equivalent statement is true for column assignments. The following lemma[17] formally states this separability property.

Lemma 5.5 (Decomposition Lemma) *For a fixed co-clustering (ρ, γ) and a fixed set of (non necessarily optimal) Lagrange multipliers Λ, and $\tilde{Z} = \zeta(\rho, \gamma, \Lambda)$, we can write*

[17] For a proof consult Banerjee et al. (2007).

$$\begin{aligned}\mathbb{E}[d_\phi(Z, \tilde{Z})] = \\ &= \mathbb{E}_U[\mathbb{E}_{V|U}[\xi(U, \rho(U), V, \gamma(V))]] = \\ &= \mathbb{E}_V[\mathbb{E}_{U|V}[\xi(U, \rho(U), V, \gamma(V))]]\end{aligned} \tag{5.27}$$

where $\xi(U, \rho(U), V, \gamma(V)) = d_\phi(Z, \tilde{Z})$.

5.4.6.3 Updating row and column clusters

Suppose we are in step $2(b)$ outlined in subsubsection 5.4.6.1. Updating the row clustering keeping the column clustering and the Lagrange multipliers fixed leads to a new value for the Bregman co-clustering objective function. Now making use of the separability property in 5.5, we can efficiently optimize the contribution of each row assignment to the overall objective function to obtain the following row cluster update step.

Lemma 5.6 *Let* ρ^{t+1} *be defined as*

$$\forall u = 1, 2, \cdots, m,\ \ \rho^{t+1}(u) = \arg\min_{1 \le g \le k} \mathbb{E}_{V|u}[\xi(u, g, V, \gamma^t(V))] \tag{5.28}$$

and let $\tilde{Z}^t = \zeta(\rho^{t+1}, \gamma^t, \Lambda^{*^t})$. *Then,*

$$\mathbb{E}[d_\phi(Z, \tilde{Z}^t)] \le \mathbb{E}[d_\phi(Z, \hat{Z}^t)] \tag{5.29}$$

where $\hat{Z}^t = \zeta(\rho^t, \gamma^t, \Lambda^{*^t})$

A similar argument applies to the step $2(c)$ where we seek to update the column clustering keeping the row clustering fixed.

Lemma 5.7 *Let* γ^{t+1} *be defined as*

$$\forall v = 1, 2, \cdots, n,\ \ \gamma^{t+1}(u) = \arg\min_{1 \le h \le l} \mathbb{E}_{U|v}[\xi(U, \rho^t(U), v, h)] \tag{5.30}$$

and let $\tilde{Z}^t = \zeta(\rho^t, \gamma^{t+1}, \Lambda^{*^t})$. *Then,*

$$\mathbb{E}[d_\phi(Z, \tilde{Z}^t)] \le \mathbb{E}[d_\phi(Z, \hat{Z}^t)] \tag{5.31}$$

where $\hat{Z}^t = \zeta(\rho^t, \gamma^t, \Lambda^{*^t})$

After such updates, the approximation $\tilde{Z}^t$ is closer to the original Z than the earlier Bregman information solution $\hat{Z}^t$, but the Lagrange multipliers Λ^{*^t} are not optimal, in general. In other words, the approximation $\tilde{Z}^t$ is not an MBI solution. For the given co-clustering $(\rho^{t+1}, \gamma^{t+1})$, let $\Lambda^{*^{t+1}}$ be the optimal Lagrange multipliers for the corresponding MBI problem in Equation 5.20. The corresponding approximation $\hat{Z}^{t+1} = \zeta(\rho^{t+1}, \gamma^{t+1}, \Lambda^{*^{t+1}})$ is an MBI solution.

Algorithm 3 Bregman generalized hard co-clustering algorithm

procedure GENERALIZEDBREGMANHCC($\mathbf{Z}, w, d_\phi, k, l$)
  $(\rho, \gamma) \leftarrow$ initialize($k, l, \mathbf{Z}$) ▷ Several policies. Simplest: random
  repeat
    ▷ Update MBI solution ($\Lambda*$)
    $\Lambda^* \leftarrow \arg\max_\Lambda L(\Lambda)$ ▷ $L(\cdot)$ is Lagrangian dual of the MBI problem
    ▷ Update row clusters (ρ)
    for $u \leftarrow 1, m$ **do**
      $\rho(u) \leftarrow \arg\min_{1 \leq g \leq k} \mathbb{E}_{V|u}[\xi(u, g, V, \gamma(V))]$
      where $\xi(U, \rho'(U), V, \gamma(V)) = d_\phi(Z, \tilde{Z})$ and $\tilde{Z} = \zeta(\rho', \gamma, \Lambda^*)$ for any ρ'
    end for
    ▷ Update column clusters (γ)
    for $v \leftarrow 1, n$ **do**
      $\gamma(v) \leftarrow \arg\min_{1 \leq h \leq l} \mathbb{E}_{U|v}[\xi(U, \rho(U), v, h)]$
      where $\xi(U, \rho(U), V, \gamma'(V)) = d_\phi(Z, \tilde{Z})$ and $\tilde{Z} = \zeta(\rho, \gamma', \Lambda^*)$ for any γ'
    end for
  until convergence
  return (ρ, γ)
end procedure

Lemma 5.8 *Let $\hat{Z}^{t+1} = \zeta(\rho^{t+1}, \gamma^{t+1}, \Lambda^{*^{t+1}})$ be the minimum Bregman information solution corresponding to the co-clustering $(\rho^{t+1}, \gamma^{t+1})$ with $\Lambda^{*^{t+1}}$ being the optimal Lagrange multipliers for Equation 5.20. Then*

$$\mathbb{E}[d_\phi(Z, \hat{Z}^{t+1})] \leq \mathbb{E}[d_\phi(Z, \tilde{Z}^t)] \tag{5.32}$$

where $\tilde{Z}^t = \zeta(\rho^{t+1}, \gamma^{t+1}, \Lambda^{*^t})$

The proof is in Banerjee et al. (2007).

5.4.6.4 The algorithm

The meta algorithm for generalized Bregman hard co-clustering is based on the idea previously outlined and on lemmas and theorems exposed. Such an algorithm is guaranteed to achieve local optimality.

Theorem 5.4 *The general Bregman co-clustering algorithm (see Algorithm 3) converges to a solution which is locally optimal for the Bregman co-clustering problem of Equation 5.26, i.e. the objective function cannot be improved by changing either the row clustering, the column clustering or the Lagrange multipliers.*

The proof is in Banerjee et al. (2007).
Note that updating Λ is the same as obtaining the MBI solution. When the Bregman divergence is I-divergence or squared deviation, the minimum Bregman information problem has an analytic closed form solution for any co-clustering basis, as shown in Table 5.2 and Table 5.3. Hence, it is straightforward to obtain the

row and column cluster update steps and implement these Bregman co-clustering algorithm instances. The resulting algorithms involve computational effort that is linear per iteration in the size of the data and are hence scalable. In general, the MBI problem has a unique solution since it involves a strictly convex objective function and linear constraints.
However, the solution need not have a closed form and has to be obtained numerically using iterative projection algorithms, which in turn involves solving nonlinear systems of equations. In the general case, the Bregman co-clustering algorithm will include such iterative projection procedures as a sub-routine.
Hence, since finding the solution for the MBI problem with no closed form solution is a crucial step, a couple of iterative algorithms is available. The first one is the *Bregman's algorithm* originally proposed by Bregman (1967); the other one is the *Iterative scaling algorithm* proposed by Della Pietra et al. (2001). Both of them are detailedly reviewed in Banerjee et al. (2007).
The instantiation of the meta algorithm to Itakura-Saito distance is an example of instance which does not have a closed form solution for the related MBI problem instance.

5.5 Bregman Block Average Co-clustering

The *Bregman Block Average Co-clustering* is an important special case of the Bregman hard co-clustering where the summary statistics are derived by aggregating along the co-clusters, i.e. the summary statistics preserved are just the co-cluster means. Hence, in this case, for a given co-clustering (ρ, γ), $\hat{Z}$ has to be reconstructed based only on the co-cluster means. It is easy to see that this situation equals to choosing the co-clustering basis $\mathcal{C}_2$.
The peculiarity of the co-clustering basis $\mathcal{C}_2$ is that for any Bregman divergence, the related MBI problem instance always has a closed form solution. This result implies that we can use any Bregman divergence with the co-clustering basis $\mathcal{C}_2$ and be sure that algorithms involve a computational effort that is linear per iteration in the size of the data and are hence scalable.

5.5.1 Minimum Bregman Information

In the general case (see subsection 5.4.4) the best approximation $\hat{Z}$ is picked up among all $Z^{'}$ that preserve all the summary statistics relevant to the co-clusteirng basis chosen. In the case of the block average co-clustering, the search for MBI solution is restricted to all $Z^{'}$ which only preserve the co-cluster means. Therefore, we can write $\mathcal{S}_A$ as follows

$$\mathcal{S}_A = \{Z^{'} | \mathbb{E}[Z|\hat{u}, \hat{v}] = \mathbb{E}[Z^{'}|\hat{u}, \hat{v}], \forall \hat{u}, \hat{v}\} \tag{5.33}$$

The MBI problem statement does not change

$$\hat{Z}_A \equiv \arg \min_{Z' \in \mathcal{S}_A} I_\phi(Z^{'}) \tag{5.34}$$

but in this case the solution is unique and is given by

$$\hat{Z}_A = \mathbb{E}[Z|\hat{U}, \hat{V}].$$

The uniqueness is proved in Banerjee et al. (2007). Such a result means that for every Bregman divergence, the MBI solution for the co-clustering basis $\mathcal{C}_2$ is always the same, as we can see in Table 5.2 and Table 5.3.

5.5.2 Problem formulation

The problem formulation (see subsection 5.4.5) does not change either

$$(\rho^*, \gamma^*) = \arg\min_{(\rho,\gamma)} \mathbb{E}[d_\phi(Z, \hat{Z})] = \arg\min_{(\rho,\gamma)} (I_\phi(Z) - I_\phi(\hat{Z})) = \arg\max_{(\rho,\gamma)} I_\phi(\hat{Z}) \quad (5.35)$$

Anyway, in the special case of the block average co-clustering, the reconstruction from the MBI solution $\hat{Z}$ is also the best approximation of the original Z among all functions of the co-cluster means, so we have an equivalent formulation for the co-clustering problem in this case

$$(\rho^*, \gamma^*) = \arg\max_{(\rho,\gamma)} \min_{Z' \in \mathcal{S}_A} I_\phi(Z') \quad (5.36)$$

Hence the best co-clustering is the one that results in the matrix reconstruction corresponding to the minimum approximation error, or equivalently, the one that solves the max-min problem above.

5.5.3 Towards the algorithm

In this section we show that the previously presented results, such as the decomposition lemma, still hold. Moreover, we build a less general meta algorithm useful for all instances of the block average co-clustering. As we will see in the sequel, such an algorithm is a direct generalization of the algorithm for the Bregman (one-way) clustering (see Algorithm 2).

5.5.3.1 Separability property

The separation property in subsubsection 5.4.6.2 still holds. In the specific case, we have the following particular form for Equation 5.27

$$\begin{aligned} \mathbb{E}[d_\phi(Z, \hat{Z})] &= \sum_{u,v} w_{uv} d_\phi(z_{uv}, \overline{z}_{\rho(u)\gamma(v)}) = \\ &= \sum_{g=1}^{k} \sum_{h=1}^{l} \sum_{u:\ g=\rho(u)} \sum_{v:\ h=\gamma(v)} \sum_{u,v} w_{uv} d_\phi(z_{uv}, \overline{z}_{gh}) \end{aligned} \quad (5.37)$$

where $\overline{z}_{\hat{u}\hat{v}} = \mathbb{E}[Z|\hat{u}, \hat{v}] = \overline{z}_{\rho(u)\gamma(v)} = \hat{z}_{uv}$ is a co-cluster mean.
So, we have decomposed the objective function of in terms of row cluster assignment and column cluster assignment.

Algorithm 4 Bregman block average co-clustering algorithm

procedure BREGMANBLOCKCC($\mathbf{Z}, w, d_\phi, k, l$)
  $(\rho, \gamma) \leftarrow$ initialize($k, l, \mathbf{Z}$) ▷ Several policies. Simplest: random
  repeat
    ▷ Update co-cluster means (the MBI solution in this case)
    for $g \leftarrow 1, k$ **do**
      for $h \leftarrow 1, l$ **do**
        $\overline{z}_{gh} = \frac{\sum_{u:\rho(u)=g} \sum_{v:\gamma(v)=h} w_{uv} z_{uv}}{\sum_{u:\rho(u)=g} \sum_{v:\gamma(v)=h} w_{uv}}$ ▷ normalized means
      end for
    end for
    ▷ Update row clusters (ρ)
    for $u \leftarrow 1, m$ **do**
      $\rho(u) \leftarrow \arg\min_{1 \le g \le k} \sum_{h=1}^{l} \sum_{v:\ \gamma(v)=h} w_{uv} d_\phi(z_{uv}, \overline{z}_{\rho(u)h})$
    end for
    ▷ Update column clusters (γ)
    for $v \leftarrow 1, n$ **do**
      $\gamma(v) \leftarrow \arg\min_{1 \le h \le l} \sum_{g=1}^{k} \sum_{u:\ \rho(u)=g} w_{uv} d_\phi(z_{uv}, \overline{z}_{g\gamma(v)})$
    end for
  until convergence
  return (ρ, γ)
end procedure

5.5.3.2 Updating row and column clusters

We can provide a specific formulation also for updating the row and column clusters. The formula in Lemma 5.6 assumes the following form in the case of the block average co-clustering

$$\forall u = 1, 2, \cdots, m,\ \ \rho^{t+1}(u) = \arg\min_{\rho(u)} \sum_{h=1}^{l} \sum_{v:\ \gamma(v)=h} w_{uv} d_\phi(z_{uv}, \overline{z}_{\rho(u)h}) \tag{5.38}$$

The formula in Lemma 5.7 assumes the following form

$$\forall v = 1, 2, \cdots, n,\ \ \gamma^{t+1}(v) = \arg\min_{\gamma(v)} \sum_{g=1}^{k} \sum_{u:\ \rho(u)=g} w_{uv} d_\phi(z_{uv}, \overline{z}_{g\gamma(v)}) \tag{5.39}$$

5.5.3.3 The algorithm

Since the MBI solution is always the co-clusters means, and Algorithm 4 essentially alternates between updating the row and column assignments, and updating the co-cluster means, the block average co-clustering algorithm is a direct generalization of the algorithm for the Bregman (one-way) clustering (see Algorithm 2). As usual, it starts with an an arbitrary choice of co-clustering (ρ, γ). At every iteration, either the row clustering or the column clustering is updated in

order to decrease the objective function. In practice, one could run multiple iterations of such updates. After the assignments have been updated for all rows and columns, the co-clustering means are updated, which further decreases the objective. The process is repeated till convergence. Since the objective decreases at every iteration, and the objective is lower bounded, the algorithm is guaranteed to converge to a (local) minimum of the objective, like in the case of the meta algorithm for generalized Bregman hard co-clustering (see subsubsection 5.4.6.4).

5.5.4 Block average co-clustering as Matrix Factorization

The block average co-clustering algorithm (see Algorithm 4) can also be viewed as solving a matrix factorization problem.
Let $\mathbf{Z}$ be the $m \times n$ matrix corresponding to the random variable Z and $\mathbf{W} \in \mathbb{R}_+^{m \times n}$ denote the matrix corresponding to a probability measure over the matrix elements. Let $\mathbf{R} \in \{0,1\}^{m \times k}$ and $\mathbf{C} \in \{0,1\}^{n \times l}$ denote the row and column cluster membership matrices, i.e.

$$r_{ug} = \begin{cases} 1 & g = \rho(u) \\ 0 & \text{otherwise} \end{cases}, \quad c_{vh} = \begin{cases} 1 & h = \gamma(v) \\ 0 & \text{otherwise} \end{cases}$$

Further, let $\mathbf{M}$ be a $k \times l$ matrix corresponding to the co-cluster means, i.e. expectations or weighted averages of the matrix values over the co-clusters. Since the minimum Bregman information solution for the block co-clustering case are the co-cluster averages, the reconstructed matrix $\hat{\mathbf{Z}}$ can be expressed as the product $\mathbf{RMC}^T$. Therefore, the co-clustering problem is essentially reduced to finding row assignment matrix $\mathbf{R}$, column assignment matrix $\mathbf{C}$ such that the approximation error $d_{\phi_w(\mathbf{Z},\hat{\mathbf{Z}})}$ is minimized where $\hat{\mathbf{Z}} = \mathbf{RMC}^T$.

5.6 Computational complexity

According to the MBI solution form (closed or not), we distinguish two cases for calculating the computational complexity. As already stated in subsubsection 5.4.6.4, when the instance of the MBI problem at hand has a closed form solution,[18] the resulting algorithm involves a computational effort that is linear per iteration in the size of the data and are hence scalable, because the complexity of the MBI solution computation can be considered $O(1)$. In other words, in this case it is enough to calculate the computational complexity of the Bregman block average co-clustering (see Algorithm 4). We can do that in the same way we have already done for the K-means (see subsection 4.1.1). Let I be the number of iterations, let k and l be the number of row clusters and column clusters respectively, let d be the dimensionality of the space and let m and n be the number of rows and

[18] We recall that this is definitely true for (i) the co-clustering basis $\mathcal{C}_2$ and all Bregman divergences; (ii) for the I-divergence and all co-clustering bases; (iii) for the Euclidean distance and all co-clustering bases.

columns of a data matrix respectively. The worst-case running time complexity in the case of a closed form MBI solution is $O(I(km + ln)d)$.
When we deal with an instance that leads to a non-closed form MBI solution (e.g. with the Itakura-Saito distance), we have to consider also the complexity of the algorithm we have chosen to analytically compute the MBI solution (see subsubsection 5.4.6.4), i.e. either the complexity of the *Bregman's Algorithm* (Bregman, 1967) or the complexity of the *iterative scaling algorithm* (Della Pietra et al., 2001).

5.7 Obtaining previous works

In this section we show how it is possible to obtain well know clustering algorithms as instances of the Bregman hard co-clustering meta algorithm. In particular we show this fact for the K-means (see section 4.1), the Information-Theoretic Co-clustering (Dhillon et al., 2003b) and the two Minimum Sum Squared residue Co-clustering algorithms in Cho et al. (2004a).

5.7.1 K-means

For obtaining the K-means from the Bregman co-clustering it is necessary to choose the $\mathcal{C}_2$ as co-clustering basis and the squared Euclidean distance as Bregman divergence. Moreover, we must fix the number of column cluster l to the number of columns n, i.e. we have not to perform any feature clustering. Since the K-means relies on the co-clustering basis $\mathcal{C}_2$, it actually is an instance of the Bregman block average co-clustering. In fact, if we put $l = n$, the block average co-clustering equals to the Bregman one-way clustering algorithm. If we use the squared Euclidean distance in the latter, we exactly have the K-means algorithm.
Let us provide some mathematical details. The Bregman co-clustering objective function in this case is

$$\mathbb{E}_w[(Z - \hat{Z})^2] = \sum_{u=1}^{m}\sum_{v=1}^{n} w_{uv}\|z_{uv} - \hat{z}_{uv}\|^2 \tag{5.40}$$

In the case of the K-means, all elements are equiprobable, so

$$\forall u = 1, 2, \cdots, m, v = 1, 2, \cdots, n, \;\; w_{uv} = \frac{1}{n}$$

By replacing this in Equation 5.40, we have

$$\frac{1}{n}\sum_{u=1}^{m}\sum_{v=1}^{n} \|z_{uv} - \hat{z}_{uv}\|^2 \tag{5.41}$$

which equals to the K-means objective function.

5.7.2 Minimum Sum Squared Residue Co-clustering

In Cho et al. (2004a) there are two different co-clustering algorithms based on the squared Euclidean distance, depending on the definition of the *squared residue*.

The first one essentially is the K-means algorithm with the "feature clustering step" enabled (Cho et al., 2004a, eq. 3.1). So, the co-clustering basis and the Bregman divergence are the same chosen for the K-means. Even the objective functions is the same. The difference between K-means and the first variation of the minimum sum squared residue co-clustering is just the "feature clustering step".
The second variation of the minimum sum squared residue co-clustering (Cho et al., 2004a, eq. 3.2) can be obtained from the Bregman hard co-clustering by employing the co-clustering basis $\mathcal{C}_6$ and, obviously, the squared Euclidean distance. In such a case, the objective function is $\mathbb{E}[(Z-\mathbb{E}[Z|U,\hat{V}]-\mathbb{E}[Z|\hat{U},V]+\mathbb{E}[Z|\hat{U},\hat{V}])^2]$. The same objective function also holds for the work presented in Cheng and Church (2000), but in this case another type of algorithm is used which cannot be obtained by instantiating the Bregman hard co-clustering algorithm.

5.7.3 Information-Theoretic Co-clustering

The information-theoretic formulation of the co-clustering is one of the main reason why the research which has led to Bregman co-clustering began. In fact, we can notice much similarities in the two works, especially in the problem formulation viewpoint. In the information-theoretic framework the problem is formulated as the minimization of the loss in mutual information $I(\cdot)$

$$(\rho^*, \gamma^*) = \arg\min_{(\rho,\gamma)}(I(U;V) - I(\hat{U};\hat{V})) = \arg\max_{(\rho,\gamma)} I(\hat{U};\hat{V}) \tag{5.42}$$

which is very similar to Equation 5.26.
The matrix $\mathbf{Z}$ is interpreted as a contingency table or, equivalently, as an empirical joint probability distribution of two discrete random variables. Let $p(X,Y)$ denote such a joint probability distribution. The problem in Equation 5.42 actually equals to the problem of finding another joint probability distribution

$$q(X,Y) = p(\hat{X},\hat{Y})p(X|\hat{X})p(Y|\hat{Y})$$

such that $\mathrm{KL}(p(X,Y)\|q(X,Y))$ is minimized, where KL is the KL-divergence (or relative entropy). It can be shown that the probability distribution $q(\cdot)$ which minimizes such a divergence has the maximum Shannon entropy (Banerjee et al., 2004a; Dhillon et al., 2003b).
If we choose the I-divergence (a generalized form of the KL-divergence) for the Bregman hard co-clustering problem, we obtain the objective function

$$\mathbb{E}[Z\log(Z/\hat{Z}) - Z + \hat{Z}] = \mathbb{E}[Z\log(Z/\hat{Z})] \text{ since } \mathbb{E}[Z] = \mathbb{E}[\hat{Z}].$$

For the co-clustering basis $\mathcal{C}_5$ and Z based on a joint distribution, this reduces to $\mathrm{KL}(p(\cdot)\|q(\cdot))$, where $q(\cdot)$ is the joint distribution corresponding to the minimum Bregman solution. Since the MBI principle equals the Maximum-Entropy principle (see Example 5.6), the q is the joint distribution has the maximum Shannon entropy, as stated above.

5.8 Novel applications

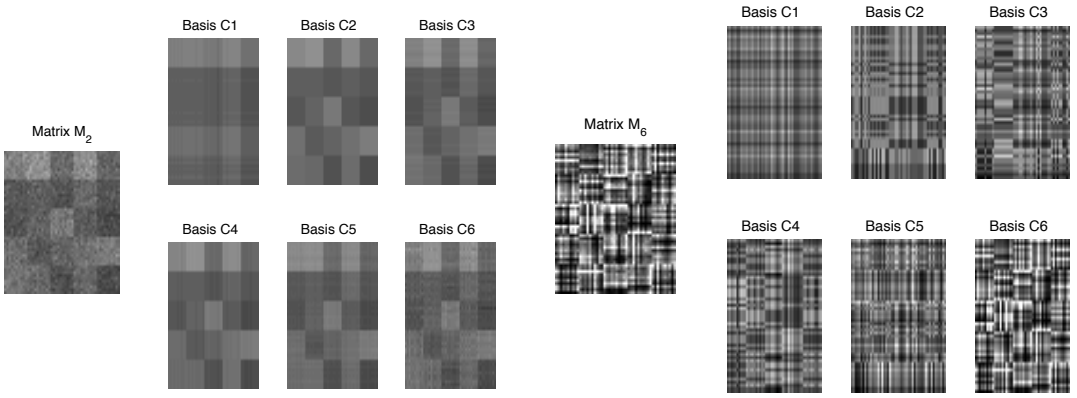
The Bregman framework for the hard co-clustering carries some interesting novel applications with it, such as the matrix approximation (or compression) and the missing values prediction (George and Merugu, 2005; Huynh and Vu, 2007; Waters and Raghavan, 2007).

5.8.1 Matrix approximation

A direct consequence of the Bregman co-clustering problem formulation is the possibility of exploiting the matrix approximation behavior. The formulation allows approximation models of various complexities depending on the statistics that are constrained to be preserved. There are two key advantages to this flexibility. First, preserving summary statistics of the data may be crucial for some applications as well as important for subsequent analysis. Since the statistics preserving property is intrinsic to this approach, it is readily applicable to domains where summary statistics are important. Second, the multiple sets of preserved statistics may enable discovery of different structural patterns in the data.
Besides, unlike existing methods, these approximation techniques can work with general divergence functions and since the methods are iterative and do not involve eigenvalue computations, they are significantly faster than other well-known approximation methods. Hence, they are more appropriate for large data matrices.
It is clear that we can look to the matrix approximation as a compression algorithm that is capable of preserving the specified summary statistics. When these specified statistics capture the natural structure of the data, it is possible to obtain a very accurate few-parameter representation of the original data (see Figure 5.2).

5.8.2 Missing values prediction

The problem of missing values arises in multiple application domains, as we already stated in section 3.7. Anyway, in addition to having a clustering algorithm which is robust with respect to the missing values, we can also be interested in the estimation of the missing values. Such an exigency is frequent in several application domains, such as recommender systems, bioinformatics, computational auditory scene analysis, etc. We found common cases in recommender systems, for example the prediction of the preference of a given user for a given item using known preferences of the other users. One of the earliest and most popular approaches to solve this problem is by computing the Pearson correlation of each user with all other users based on the known preferences and predict the unknown rating by proportionately combining all the users' ratings. Based on the observation that the known ratings correspond to elements in a matrix and the missing ratings can be predicted using suitable low parameter approximations of the ratings matrix, a number of other collaborative filtering approaches based on matrix approximation methods have been proposed in recent years.



(a) Simple matrix (b) Fairly complex matrix

Figure 5.2: ***(a)****: Co-clustering-based approximation of a simple* 50×50 *synthetic matrix (*$\mathbf{M}_2$*) using various co-clustering bases, squared distortion and* $k = l = 5$*. While the matrix is too complex for* $\mathcal{C}_1$*, all bases from* $\mathcal{C}_2$ *onwards get an accurate approximation.* ***(b)****: Co-clustering-based approximation of a simple* 50×50 *synthetic matrix (*$\mathbf{M}_6$*) using various co-clustering bases, squared distortion and* $k = l = 5$*. In this case the original matrix is fairly complex and only* $\mathcal{C}_6$ *gets an accurate approximation. All other schemes have more errors, with the simple bases (*$\mathcal{C}_1$ *and* $\mathcal{C}_2$*) having high errors. Note that all matrices are shown with a consistent permutation (which the co-clustering finds) for easy visual comparison (Banerjee et al., 2007, fig. 3, 4).*

Since the Bregman hard co-clustering problem relies on a matrix approximation problem, it fits well the problem of the missing values prediction. Very promising results was obtained in the movie ratings prediction (Banerjee et al., 2007; Huynh and Vu, 2007), where the estimation based on the Bregman matrix approximation compares better with other well known methods which rely on different matrix approximations.
Another application of the missing values prediction based on Bregman matrix approximation has been made in the field of the *computational auditory scene analysis* also known as *machine hearing* (Waters and Raghavan, 2007). In this case the missing values estimation has been used for completing (estimating) some crucial informations in a dataset, in order to have no missing values in some data designed to become a training set for a supervised learning machine.
The missing values prediction could be an interesting application also in several fields of the science, like physics, astrophysics, chemistry.

5.9 Advantages

The first remarkable advantage of the Bregman co-clustering framework is the twofold generalization.[19] In fact, the meta algorithm can be instantiated with any Bregman divergence and any of the six co-clustering bases, so we have the first

[19] Let us recall that there is a third generalization which we are not considered in our discussion. In fact, the Bregman clustering and co-clustering frameworks can also be extended to the "soft" case, where the intersection of the clusters need not to be empty (Banerjee et al., 2005b, 2004c,

generalization: a generic alternate minimization scheme that can be used with a large class of distortion functions and can preserve different summary statistics. The second generalization is in terms of the clustering problem we are interested to solve. In fact, we can choose to perform a co-clustering, choosing $k < m$ and $l < n$, or we can prefer a simple one-way data clustering by fixing $l = n$, or we can even perform just a feature clustering by fixing $k = m$.
The direct consequence of such a generalization is the independence from the application domain. To tackle a problem in a new application domain where the well known instantiations[20] are not effective, we have to study the problem at hand and build a new Bregman divergence. The particular choice of the divergence function can be determined by (i) the data type, for example, if the data corresponds to joint probability distributions, relative entropy is an appropriate choice as the divergence function; (ii) the appropriate noise model, for example, Euclidean distance is appropriate for Gaussian noise, Itakura-Saito is appropriate for Poisson noise, etc.; or (iii) domain knowledge/requirements, for example, sparsity of the original matrix can be preserved using I-divergence. Afterwards, we have to choose what summary statistics retain, i.e. we have to select the most appropriate co-clustering basis. This can be empirically done by performing a set of pilot experiments with all co-clustering bases and observe which is the best one (Banerjee et al., 2007, sec. 6.2).
Furthermore, the Bregman co-clustering framework has the intrinsic characteristic of a co-clustering algorithm: the ability to exploit the clear duality between rows and columns, enabling us to find similar objects and their interplay with feature clusters. For example, you need only to consider a text mining problem which consists of grouping documents and revealing what are the words which they was clustered up on.
Finally, the novel applications we have discussed in section 5.8 are doubtless interesting advantages of this clustering technique.

5.9.1 Addressed issues

Among the clustering issues discussed in chapter 3, the Bregman co-clustering addresses the problem of the data matrices sparseness, i.e. it is robust with respect to the missing values. Moreover, it also addresses the high dimensionality problem and is robust with respect to the noise. All these issues are addressed thanks to the ability of the implicit dimensionality reduction.
Dimensionality reduction techniques are widely used in some application domains, such as text clustering, to handle sparsity and high-dimensionality. Typically, the dimensionality reduction step comes before the clustering step, and the two steps are almost independent (see subsection 3.6.1). In practice, is is not clear which dimensionality reduction technique to use in order to get a good clustering. The co-clustering has the interesting capability of *interleaving* dimensionality reduction and clustering. This implicit dimensionality reduction often results in

2005c; Shafiei and Milios, 2006).
[20] Euclidean distance, I-divergence, Itakura-Saito distance.

superior results when we deal with sparse and high dimensional data (Dhillon and Guan, 2003a,b; Dhillon et al., 2003b).
The sparseness of data matrix is also known as the missing values problem, especially in scientific application fields. The missing values problem can be considered a special case of the noise problem.[21] Both the noise problem and the missing values one is addressed by means of the implicit dimensionality reduction, because the "feature compression" behavior of the Bregman co-clustering also reduces the influence of the noise and the missing values.
A key peculiarity which further improves the ability of dealing with missing values, is the random variables based formulation. The possibility of specifying a probability measure other than the equiprobable one,[22] allows to assign a zero weight to missing values.

5.10 Disadvantages

Up to now we have tacitly assumed an arbitrary co-clustering initialization, in the same way this was already done for the centroid initialization in the classic version of the K-means algorithm. However, we know that this behavior is a general drawback of such alternate minimization schemes, because the clustering results become highly dependent on the random initialization. As in the case of the K-means, some heuristics for a better initialization were developed. An example of such alternative initializations in the case of the co-clustering is the *spectral approximation* algorithm for the initialization (Cho et al., 2004a), and the *farthest apart assignment* (Cho et al., 2004b), but they are not valid for all Bregman co-clustering instances.
The other disadvantage inherited from the K-means algorithm is the *poor local minima* problem. This is highly correlated to the initialization problem; in fact, if we choose an initial co-clustering with low informative content, the probability of a poor local minimum in high. In order to escape local minimum, an interesting *local search* strategy for updating clusters was developed (Cho et al., 2004a; Dhillon and Guan, 2003a,b). However, this solution increases the overall computational complexity.
The two major problems of K-means-like algorithms (hence for Bregman hard co-clustering framework) is the estimation of the number of clusters (see section 3.4) and the non-robustness with respect to the *outliers*,[23] which can negatively condition the clustering results, leading to a very trivial partitioning.

[21] We recall that in this thesis we call noise the disturbance on feature values (see section 3.6).
[22] A novel characteristic of this co-clustering framework.
[23] We recall that in this thesis we call outliers those points which do not belong to any cluster (see section 3.8).

5.10.1 Recently addressed issues

An extension of the Bregman framework, called *Bregman Bubble (Co)-clustering* was very recently presented (Deodhar et al., 2007a,b).[24] The Bregman Bubble Co-clustering framework is the ultimate generalization of the Bregman framework for clustering problems. It is able to work on multiple dense regions and consider the regions with low density as outlier points. Furthermore, it is also capable of efficiently detecting arbitrarily positioned, possibly overlapping co-clusters in a dataset and by means of a novel model selection strategy it also automatically determines the appropriate number of co-clusters. The latter capability solves the last big issue of the relocation-based algorithms: the necessity of providing the number of clusters in input.

5.11 Conclusion

In this chapter we presented a powerful generalization of the classical K-means-like strategies for the clustering. The number and the variety of the advantages is very interesting and the number of drawbacks are still decreasing, thanks to very recent contributions like the Bregman Bubble Co-clustering. We can consider this approach the state-of-the-art of the relocation based schemes for clustering, a worthy descendant of the K-means.
Notwithstanding the great level of generalization achieved by this framework, the job in this direction is not yet complete. The analysis of co-clustering has focused on data matrices that represent the relationship between two entities. Many emerging application domains collect data on relationships between multiple entities, which can be represented as a tensor. The Bregman Co-clustering can be extended to this general setting involving tensors unlike other methods that are specific to matrices. It will be worthwhile to investigate how the extensions of co-clustering to tensor data perform compared to existing techniques. In particular, several practical problem domains have known statistical dependency relationships between the several entities of interest. One of the key challenges of an extension of co-clustering to such multi-entity relational domains is to come up with efficient algorithms that take advantage of the statistical relationships and maintain succinct representations of the entities and their relationships.
The Bregman framework abstraction process also leads to an interesting result from an information theory viewpoint: the *Minimum Bregman Information (MBI) principle*. Since the approximations obtained from the MBI principle extend some of the desirable properties of the maximum entropy models to settings where a Bregman divergence other than the relative entropy is more appropriate, we get a new class of statistical modeling techniques that are applicable to more general settings. The MBI principle has potential applications beyond the clustering problems we have discussed.
Moreover, there are other additional properties of the Bregman divergences that

[24] It arise from the union of the Bregman co-clustering and the Bubble clustering by Gupta (2006).

are not remarkable for the clustering problems; on the contrary, they are very attractive from a strictly mathematical viewpoint (Banerjee et al., 2007, Appendix B).

CHAPTER

SIX

Support Vector Clustering

«I would like to demonstrate that in this area of science a good old principle is valid: "Nothing is more practical than a good theory".»

VLADIMIR NAUMOVICH VAPNIK
(VAPNIK, 1995, PREFACE)

THE Support Vector Machines (SVMs) were born as a set of related supervised learning methods used for classification and regression. SVMs were firstly used in an unsupervised way to estimate the support of high dimensional distribution (Schölkopf et al., 1999). Successively, SVM formulations for *One-class classication*[1] were provided (Schölkopf et al., 2000b; Tax, 2001; Tax and Duin, 1999a,b, 2004). Later, the use of SVMs was introduced in the clustering research.

Currently there are only three remarkable approaches in the support vector methods for clustering. The first one is called *Support Vector Clustering (SVC)* (Ben-Hur et al., 2000a,b, 2001, 2000c) and is the main subject of the research in support vector methods for clustering (Hansen et al., 2007; Hao et al., 2007; Hartono and Widyanto, 2007; Lee and Lee, 2005; Lee and Daniels, 2005a,b, 2006; Ling et al., 2006; Ling and Zhou, 2006; Nath and Shevade, 2006; Park et al., 2004; Ping et al., 2005; Puma-Villanueva et al., 2005; Ribeiro, 2002; Yang et al., 2002; Yankov et al., 2007; Zheng et al., 2006). SVC has already been tested in specific application domains, like spatial data analysis (Ban and Abe, 2004), medical imagery (Garcia and Moreno, 2004; Wang et al., 2005), content-based image retrieval (Tung and Hsu, 2005) or market research (Huang et al., 2007).

[1] Already mentioned as *novelty detection* in subsubsection 2.2.1.2 and also known as *outliers detection*, *anomalies detection*, *density estimation*, *distribution estimation*, *domain description*.

The second support vector method for clustering is a modification of the SVC, called *Multi-sphere Support Vector Clustering (MSVC)* (Asharaf et al., 2006; Chiang and Hao, 2003).
The third noticeable support vector method for clustering is the *Kernel Grower (KG)* (Camastra, 2004, 2005; Camastra and Verri, 2005), which adopts a K-means like strategy to find spherical-like clusters in the feature space.[2]
All support vector methods for clustering share a common basis: the *Support Vector Domain Description (SVDD)* (Tax, 2001; Tax and Duin, 1999a,b, 2004). In fact, all of them rely on finding the *Minimum Enclosing Ball (MEB)* in the feature space in the same way SVDD does, even though the latter aims at finding only a description of target data.
This chapter focuses on Support Vector Clustering (SVC) and its peculiarities and advantages over other clustering algorithms. We discuss it detailedly and present some of our contributions to this novel clustering technique. The other two support vector methods are described in chapter 7. However in this thesis we provide experiments for the SVC only.

6.1 Introduction to Support Vector Clustering

SVC is a method which allows for a systematic search for clustering solutions without making assumptions on their number or shape. SVC maps data points from data space to a higher dimensional *feature space* by using a suitable Mercer kernel (usually a Gaussian kernel), which has to be able to probe data at different scales. In the feature space we look for the smallest sphere that encloses the image of data points, the so-called Minimum Enclosing Ball (MEB). This sphere is mapped back to the data space, where it splits into a set of contours which enclose the data points. These contours are interpreted as cluster boundaries and they can have arbitrary shapes. Points enclosed by each separate contour are associated with the same cluster. The number of contours depends on the hyper-parameter that determines the scale at which data is probed: the *kernel width*. The greater the kernel width, the greater the number of contours, leading to a greater number of clusters. Since the contours can be interpreted as delineating the support of underlying probability distribution, the SVC can be viewed as an algorithm identifying valleys in this probability distribution.[3]
SVC can also deal with outliers by employing a soft margin parameter that allows the sphere in feature space not to enclose all points.[4] The smaller the soft margin parameter, the more soft are the "edge" of the sphere: for appropriately small values of this hyper-parameter we can also deal with strongly overlapping

[2]Kernel Grower (KG) is currently under further research at a university of Pechino, by a PhD student (Camastra, 2007).

[3]Because of this peculiarity, the SVC has also been compared with *Density-based Clustering Algorithms*, such as DBSCAN and OPTICS (Lee and Daniels, 2005b, sec. 2.2). Due to the cluster boundaries detection, another SVC peculiarity, it has also been compared with *Boundary Detecting Clustering Algorithms* (Lee and Daniels, 2005b, sec. 2.3).

[4]It is the same parameter that in an SVM classifier allows a soft margin for the separating hyperplane (see Equation 2.14).

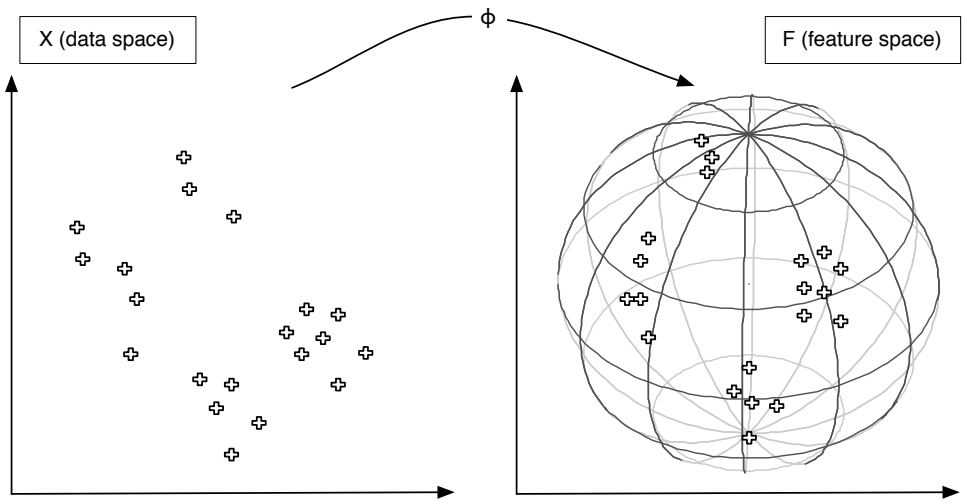


Figure 6.1: *A graphical example of a data mapping from input space to feature space using the MEB formulation.*

clusters.
Unfortunately, finding the MEB is not enough. The process of finding such a sphere is only able to detect the cluster boundaries which are modeled mapping back the sphere to the data space. This first phase was called *Cluster Description* by Ben-Hur et al. (2001). A second stage for determining the membership of points to the clusters is needed. The authors called this step *Cluster Labeling*, though it simply does a cluster assignment.[5]
In the following subsections we provide an overview of the Support Vector Clustering algorithm as originally proposed by Ben-Hur et al. (2001).

6.1.1 Cluster description

The SVM formulation for the smallest enclosing sphere was firstly presented to explain the *Vapnik-Chervonenkis (VC)* dimension (Vapnik, 1995). Later it was used for estimating the support of a high-dimensional distribution (Schölkopf et al., 1999). Finally, the MEB was used for the Support Vector Domain Description (SVDD), an SVM formulation for the *one class classification* (Tax, 2001; Tax and Duin, 1999a,b, 2004).[6] The SVDD is the basic step of the SVC and allows describing the boundaries of clusters.
Let $\mathcal{X} = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n\}$ be a dataset of n points, with $\mathcal{X} \subseteq \mathbb{R}^d$, the data space. We use a nonlinear transformation $\phi : \mathcal{X} \to \mathcal{F}$ from the input space $\mathcal{X}$ to some high dimensional feature space $\mathcal{F}$, wherein we look for the smallest enclosing sphere of radius R. This is formalized as follows

[5]The name *cluster labeling* probably descends from the originally proposed algorithm which is based on finding the connected components of a graph: the algorithms for finding the connected components usually assign the "components labels" to the vertices.

[6]An alternative SVM formulation for the same task, called *One Class SVM*, can be found in Schölkopf et al. (2000b) (see Appendix A).

$$\min_{R,\vec{a}} R^2 \tag{6.1}$$

subject to

$$\|\phi(\vec{x}_k) - \vec{a}\|^2 \leq R^2, k = 1, 2, \cdots, n$$

where $\vec{a}$ is the center of the sphere. Soft constraints are incorporated by adding slack variables ξ_k

$$\min_{R,\vec{a},\xi} R^2 + C\sum_{k=1}^{n} \xi_k \tag{6.2}$$

subject to

$$\|\phi(\vec{x}_k) - \vec{a}\|^2 \leq R^2 + \xi_k, k = 1, 2, \cdots, n$$
$$\xi_k \geq 0, k = 1, 2, \cdots, n$$

To solve this problem we introduce the Lagrangian

$$\mathcal{L}(R, \vec{a}, \xi; \beta, \mu) = R^2 - \sum_k (R^2 + \xi_k - \|\phi(\vec{x}_k) - \vec{a}\|^2)\beta_k - \sum_k \xi_k\mu_k + C\sum_k \xi_k \tag{6.3}$$

with Lagrangian multipliers $\beta_k \geq 0$ and $\mu_k \geq 0$ for $k = 1, 2, \cdots, n$. The positive real constant C provides a way to control outliers percentage, allowing the sphere in feature space to not enclose all points of the target class. As in the case of supervised SVMs (see subsection 2.6.3), this could lead to some erroneous classifications, but it allows finding a solution to the problem.
The solution is characterized by the saddle point of the Lagrangian

$$\max_{\beta,\mu} \min_{R,\vec{a},\xi} \mathcal{L}(R, \vec{a}, \xi; \beta, \mu) \tag{6.4}$$

Setting to zero the partial derivatives of $\mathcal{L}$ with respect to $R, \vec{a}$ and ξ_k, respectively, leads to

$$\sum_k \beta_k = 1 \tag{6.5}$$

$$\vec{a} = \sum_k \beta_k \phi(\vec{x}_k) \tag{6.6}$$

$$\beta_k = C - \mu_k \tag{6.7}$$

We still need to determine R. The KKT complementarity conditions[7] (Boyd and Vandenberghe, 2004, sec. 5.5.3) result in

[7] It is a generalization of the method of the Lagrange multipliers. Also in this case, if the primal problem is a convex problem, the strong duality holds.

Point	Condition	Details
SVs	$0 < \beta_k < C$	On the surface of the sphere. They describe the cluster boundaries.
BSVs	$\beta_k = C$	Outside the sphere. They allow to deal with overlapping clusters.
Internals	$\beta_k = 0$	Inside the sphere. They do not participate in cluster definition.

Table 6.1: *Classification of point images in relation to the feature space hypersphere.*

$$\xi_k \mu_k = 0 \tag{6.8}$$

$$(R^2 + \xi_k - \|\phi(\vec{x}_k) - \vec{a}\|^2)\beta_k = 0 \tag{6.9}$$

So, from the equations above we can distinguish three types of point images. It follows from Equation 6.9 that the image of a point $\vec{x}_k$ with $\xi_k > 0$ and $\beta_k > 0$ lies outside the hypersphere in feature space. Equation 6.8 states that such points have $\mu_k = 0$, hence we conclude from Equation 6.7 that $\beta_k = C$. This will be called a *Bounded Support Vector (BSV)*. A point $\vec{x}_k$ with $\xi_k = 0$ is mapped to the inside or to the surface of the hypersphere in feature space. If its β_k is such that $0 < \beta_k < C$ then Equation 6.9 implies that its image $\phi(\vec{x}_k)$ lies on the surface of the feature space sphere. Such a point will be referred to as a *Support Vector (SV)*. An SV lies on sphere surface, a BSV lies outside the sphere, and all other points lie inside it (see Table 6.1). Note that when $C \geq 1$ no BSVs exist because of the constraint in Equation 6.5.
Using these relations we may eliminate the variables R, $\vec{a}$ and μ_k, turning the Lagrangian into the Wolfe dual form that is a function of the variables β_k

$$\mathcal{W}(\beta) = \sum_k \phi(\vec{x}_k)^2 \beta_k - \sum_{k,l} \beta_k \beta_j \phi(\vec{x}_k) \cdot \phi(\vec{x}_l) \tag{6.10}$$

Since the variables μ_k do not appear in the Lagrangian they may be replaced with the constraints

$$0 \leq \beta_k \leq C, k = 1, 2, \cdots, n \tag{6.11}$$

We can replace the dot product $\phi(\vec{x}_k) \cdot \phi(\vec{x}_l)$ by a suitable Mercer kernel $K(\vec{x}_k, \vec{x}_l)$, so we can rewrite Equation 6.10 as follows

$$\mathcal{W}(\beta) = \sum_k K(\vec{x}_k, \vec{x}_k)\beta_k - \sum_{k,l} \beta_k \beta_j K i(\vec{x}_k, \vec{x}_l) \tag{6.12}$$

For each point $\vec{x}$ we can define the distance of its image in the feature space from the center of the sphere.

Definition 6.1 (Squared Feature Space Distance) . *Let $\vec{x}$ be a data point. We define the distance of its image in feature space $\mathcal{F}$, $\phi(\vec{x})$, from the center of the sphere, $\vec{a}$, as follows*

$$d_R^2(\vec{x}) = \|\phi(\vec{x}) - \vec{a}\|^2 \tag{6.13}$$

In view of Equation 6.6 and the definition of the kernel we have the *kernelized* version of the above distance

$$d_R^2(\vec{x}) = K(\vec{x},\vec{x}) - 2\sum_{k=1}^{n}\beta_k K(\vec{x}_k,\vec{x}) + \sum_{k=1}^{n}\sum_{l=1}^{n}\beta_k\beta_l K(\vec{x}_k,\vec{x}_l) \tag{6.14}$$

Since the solution vector $\vec{\beta}$ is sparse, i.e. only the Lagrangian multipliers associated to the support vectors are non-zero, we can rewrite the above equation as follows

$$d_R^2(\vec{x}) = K(\vec{x},\vec{x}) - 2\sum_{k=1}^{n_{sv}}\beta_k K(\vec{x}_k,\vec{x}) + \sum_{k=1}^{n_{sv}}\sum_{l=1}^{n_{sv}}\beta_k\beta_l K(\vec{x}_k,\vec{x}_l) \tag{6.15}$$

where n_{sv} is the number of support vectors.
It is trivial to define the radius R of the feature space sphere as the distance of any support vector from the center of the sphere

$$R = d_R(\vec{x}_k), \forall \vec{x}_k \in \mathcal{SV} \tag{6.16}$$

where $\mathcal{SV} = \{\vec{x}_k \in \mathcal{X} : 0 < \beta_k < C\}$ and $d_R(\cdot) = \sqrt{d_R^2(\cdot)}$.
The contours that enclose the points in data space are defined by the level set[8] of the function $d_R(\cdot)$

$$L_{d_R}(R) \equiv \{x \mid d_R(\vec{x}) = R\} \tag{6.17}$$

Such contours are interpreted as forming cluster boundaries. In view of Equation 6.16, we can say that SVs lie on cluster boundaries, BSVs are outside, and all other points lie inside the clusters.

6.1.1.1 The kernel choice

All works[9] about the Support Vector Clustering assume the Gaussian kernel (see subsubsection 2.6.1.1)

$$K(\vec{x},\vec{y}) = -q\|\vec{x}-\vec{y}\|^2 \overset{(q=1/2\sigma^2)}{=} -\frac{\|\vec{x}-\vec{y}\|^2}{2\sigma^2} \tag{6.18}$$

as the only kernel available. There are several motivations for the exclusive employment of this kernel. The main reason is that Tax (2001) has investigated only two kernels for the SVDD, the polynomial and the Gaussian ones. The result

[8] More on level sets in Gray (1997).

[9] Actually, in one of the preliminary papers about SVC a Laplacian kernel was also employed (Ben-Hur et al., 2000a).

was that the Gaussian kernel produces tighter descriptions (and so tighter contours in the SVC), whereas the polynomial kernel stretches the data in the high dimensional feature space, causing data to become very hard to describe with a hypersphere. In the subsubsection 6.1.3.2 we mention another important reason why the Gaussian kernel is the most common choice in this application.
In all correlated works, the employment of the Gaussian kernel has been taken for granted. Anyway, in the sequel we will also show that the SVC can work also with other types of kernels.

6.1.1.2 SVDD improvement

Tax and Juszczak (2003) have proposed a preprocessing method in order to enhance the performance of the SVDD. The SVDD (and, generally, any one-class classification formulation) is quite sensible to data distributed in subspaces which harm the performance. The preprocessing method is called *kernel whitenening* and it consists of a way to scale the data such that the variances of the data are equal in all directions.

6.1.2 Cluster labeling

The cluster description algorithm does not differentiate between points that belong to different clusters. To build a decision method for cluster membership we use a geometric approach involving $d_R(\vec{x})$, based on the following observation: given a pair of data points that belong to different clusters, any path that connects them must exit from the sphere in the feature space. Therefore, such a path contains a segment of points $\vec{y}$ such that $d_R(\vec{y}) > R$. This leads to the definition of the adjacency matrix A between all pairs of points whose images lie in or on the sphere in feature space.
Let S_{ij} be the line segment connecting $\vec{x}_i$ and $\vec{x}_j$, such that $S_{ij} = \{\vec{x}_{i+1}, \vec{x}_{i+2}, \cdots, \vec{x}_{j-2}, \vec{x}_{j-1}\}$ for all $i, j = 1, 2, \cdots, n$, then

$$A_{ij} = \begin{cases} 1 & \text{if } \forall \vec{y} \in S_{ij}, d_R(\vec{y}) \leq R \\ 0 & \text{otherwise.} \end{cases} \tag{6.19}$$

Clusters are now defined as the connected components of the graph induced by the matrix A. Checking the line segment is implemented by sampling a number m of points between the starting point and the ending point. The exactness of A_{ij} depends on the number m.
Clearly, the BSVs are unclassified by this procedure since their feature space images lie outside the enclosing sphere. One may decide either to leave them unclassified or to assign them to the cluster that they are closest to. Generally, the latter is the most appropriate choice.

6.1.3 Working with Bounded Support Vectors

The shape of enclosing contours in data space is governed by two *hyper-parameters* (or simply *parameters*): the kernel width q that determines the scale at which data

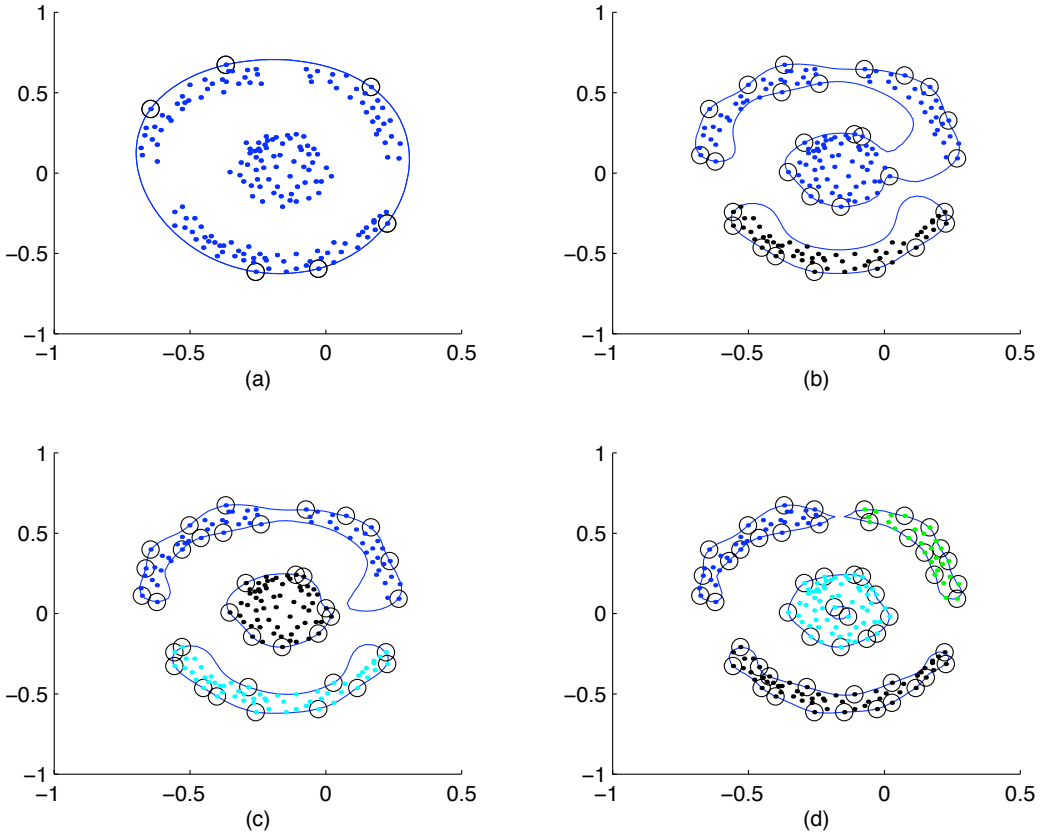


Figure 6.2: *Clustering of a dataset containing 183 points, with $C = 1$. Support vectors are designated by small circles. The figures from (a) to (d) represents the clustering executed at increasingly greater kernel width values (Ben-Hur et al., 2001, fig. 1).*

is probed, and the soft margin constant C, which allows a "soft" hypersphere in feature space, i.e. a hypersphere not enclosing all points. Therefore, with $C < 1$ some points becomes *outliers* even if they are not, i.e. they exit the hypersphere. The points that become outliers are some SVs which turn into BSVs.

The role of BSVs is twofold. A low regime of BSVs allows to deal with datasets which have *outliers* points (see section 3.8). On the other hand, a high regime of BSVs points is useful to deal with *strongly overlapping clusters*, i.e. strongly nonlinear separable problems (see subsection 2.3.1).

As stated above, the number of BSVs depends on the parameter C. From the Equation 6.5 and Equation 6.11 it follows that

$$n_{bsv} < \frac{1}{C} \tag{6.20}$$

where n_{bsv} is the number of BSVs. Thus $1/nC$ is an upper bound on the fraction of BSVs, and it could be more natural to work with the parameter

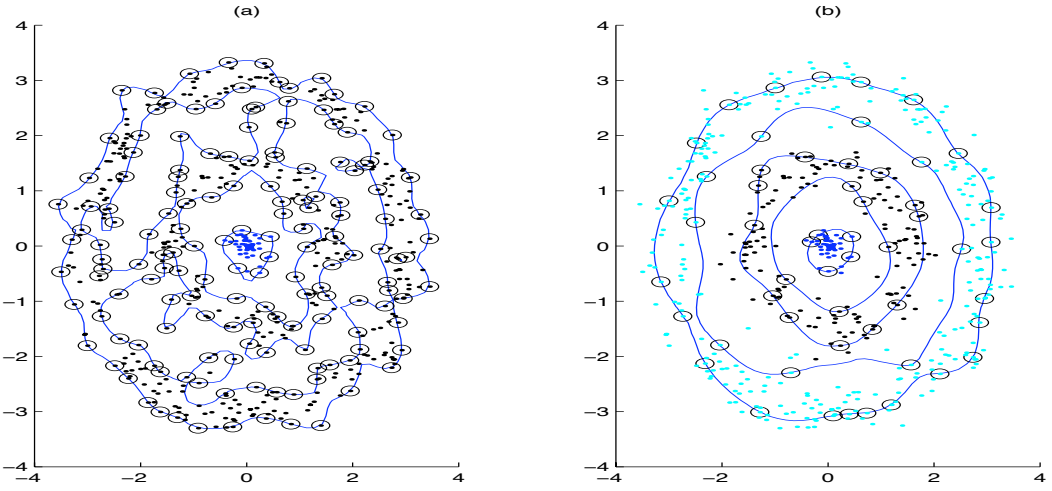


Figure 6.3: *Clustering with and without BSVs. The inner cluster is composed of* 50 *points generated from a Gaussian distribution. The two concentric rings contains 150/300 points, generated form a uniform angular distribution and a radial Gaussian distribution.* (a) *The rings cannot be distinguished when* $C = 1$*. Only the inner cluster was separated.* (b) *Outliers allow easy clustering and all clusters was separated (Ben-Hur et al., 2001, fig. 3).*

$$p = \frac{1}{nC} \tag{6.21}$$

Asymptotically, for large values of n, the fraction of BSVs tends to p, as noted in Schölkopf et al. (2000b). However, in this thesis we prefer to still use the parameter C instead, because the parameter p is too similar to the parameter ν of the *One class SVM* (Schölkopf et al., 2000b) we will introduce later for practical motivations (see Appendix A), but the meaning of the two parameters is slightly different.

6.1.3.1 Dealing with outliers

When distinct clusters are present, but some outliers prevent contour separation, it is very useful to employ BSVs. We can see this in the Figure 6.3: without BSVs contour separation does not occur for the two outer rings. When some BSVs are present, the clusters are easily separated. The difference between data that are contour-separable without BSVs and data that require use of BSVs is that the latter have an overlap (even small) in their density functions, caused by noise and/or outliers points. Such a situation is schematically illustrated in Figure 6.4.

6.1.3.2 Strongly overlapping clusters

The Support Vector Clustering is also useful in cases where clusters strongly overlap. In such a case a high BSV regime is needed. However the BSV points are different from the "real" outliers, i.e. those points which are completely foreign to

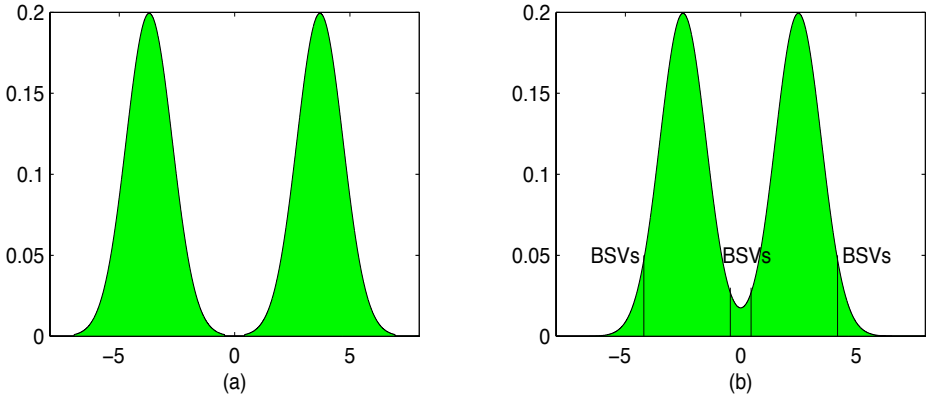


Figure 6.4: *Clusters with overlapping density functions require the introduction of BSVs (Ben-Hur et al., 2001, fig. 4).*

the problem at hand. While BSVs can still be classified in relation to the clusters detected by means of the remaining points, the "real" outliers very often form singleton clusters (or, anyway, low-cardinality clusters) which we can choose to neglect. Anyway, a high BSV regime certainly leads to some misclassifications.
In case of high BSV regime a different interpretation of the results is required. The hypersphere in the feature space have to be interpreted as representing clusters cores, rather than the envelope of all data.
Note that Equation 6.17 for the reflection of the sphere in data space can be expressed as

$$\{x \mid \sum_k \beta_k K(\vec{x}_k, \vec{x}) = \rho\} \tag{6.22}$$

where ρ is determined by the value of this sum on the support vectors. The set of points enclosed by the contour is

$$\{x \mid \sum_k \beta_k K(\vec{x}_k, \vec{x}) > \rho\} \tag{6.23}$$

In the extreme case when almost all data points are BSVs ($C \to 1/n$), the summation in the following expression

$$P_{SVC} = \sum_k \beta_k K(\vec{x}_k, \vec{x}) \tag{6.24}$$

is approximately equal to

$$P_w = \frac{1}{n} \sum_k K(\vec{x}_k, \vec{x}) \tag{6.25}$$

This last expression is known as a *Parzen window estimate* of the density function (Bishop, 2006, sec. 2.5.1).[10] In this high BSV regime, we expect the contour in

[10]Up to a normalization factor, if the kernel is not appropriately normalized.

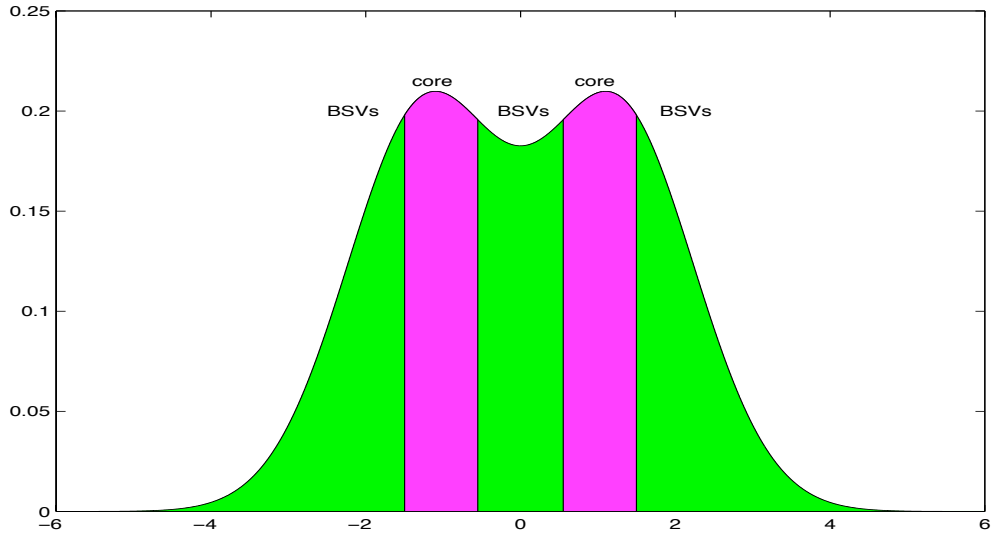


Figure 6.5: *In the case of significant overlap between clusters the SVC algorithm identifies clusters according to dense cores, or maxima of the underlying probability distribution. (Ben-Hur et al., 2001, fig. 5).*

data space to enclose a small number of points which lie near the maximum of the Parzen-estimated density. In other words, the contour specifies the *core* of the probability distribution. This is schematically drawn in Figure 6.5.
In this situation, the SVC algorithm is closely related to the scale-space algorithm proposed by Roberts (1996). He defines the cluster centers as maxima of the Parzen window estimator $P_w(\vec{x})$. The Gaussian kernel plays an important role in his analysis: it is the only kernel for which the number of maxima (hence the number of clusters) is a monotonically nondecreasing function of the kernel width q. This is the counterpart of contour splitting in SVC.
However, the SVC has some advantages over the scale-space algorithm. First, the SVC also identifies clusters represented by small regions. Besides, the computational advantage of the SVC is that, instead of solving a problem with many local maxima, it identifies core boundaries by an SV method with a global optimum solution. The conceptual advantage of the SVC is that it defines a region rather than just a peak, as the core of the cluster.

6.1.4 Execution scheme

With the Support Vector Clustering we do not need to hypothesize anything about the number of clusters, i.e. we do not have the problem of estimating the number of clusters (see section 3.4). The execution scheme of the SVC is an iterative scheme with the goal of finding the best hyper-parameters setting which, in turn, leads us to the best partitioning of the input dataset. The iterative scheme of the SVC execution can be either *divisive* or *agglomerative*.
The divisive/agglomerative scheme and the independence from the number of

clusters may lead the reader to think to the SVC as a hierarchical clustering algorithm. The authors came to the same erroneous conclusion in one of the early publications about the SVC (Ben-Hur et al., 2000a). Later, the same authors have found counterexamples when BSVs are employed, such that the strict hierarchy is not guaranteed.
However, the advantages of a hierarchical clustering algorithm still hold. We have only to decide which is the best scheme. In the case of the divisive scheme, we start with the lowest suitable kernel width value and then increase it until a stopping criterion is met. In the case of the agglomerative scheme we start from the highest suitable kernel width value and then decrease it. In both cases we have an estimate of the initial kernel width value (Tax, 2001, sec. 2.4)

$$q = \frac{1}{\max_{i,j} \|\vec{x}_i - \vec{x}_j\|^2} \tag{6.26}$$

and

$$q = \frac{1}{\min_i \|\vec{x}_i - \mathrm{NN}(\vec{x}_i)\|^2} \tag{6.27}$$

respectively.[11]
The authors chose the divisive approach for the SVC execution. The reason is quite trivial: the number of clusters produced at the highest kernel width value tends to the number of objects in the input dataset, so it is much greater than the average number of clusters we encounter in real-world problems. Therefore, by using the agglomerative scheme we perform too many iterations to reach the best partitioning.
Thus, we start from the kernel width value in Equation 6.26, which leads SVC to find a single cluster enclosing all data points. At this kernel width value no outliers are needed, so we choose $C = 1$. As q is increased, we expect to find bifurcations of clusters. In general, by keeping the number of SVs low we ensure smoother cluster boundaries. If clusters of single or few points break off, or cluster boundaries become very rough, C should be decreased, whereby many SVs may be turned into BSVs, and smooth cluster (or core) boundaries emerge. In other words, the parameters q and C should be varied along a direction that keeps the number of SVs low. A second criterion for good clustering solutions is the stability of cluster assignments over some range of the two parameters.
A first problem we have to face using a divisive approach is the decision when to stop dividing the clusters. Many approaches applicable to any clustering algorithm exist, such as Ben-Hur et al. (2002) and Bertoni and Valentini (2006). However, an *ad hoc* approach which exploits the SVC characteristics would be a better solution. A first step in this direction was moved by Ben-Hur et al. (2001) that proposed to use the number of support vectors as an indication of a meaningful solution. Hence, we should stop SVC when the fraction of SVs exceeds some predefined threshold.

[11] NN stands for Nearest Neighbor.

Algorithm 5 The general execution scheme for the SVC

```
1: procedure SVC(𝒳)
2:     q ← initial kernel width
3:     C ← initial soft constraint
4:     while stopping criterion is not met do
5:         β ← clusterDescription(𝒳, q, C)        ▷ Find the MEB via SVDD
6:         results ← clusterLabeling(𝒳, β)
7:         choose new q and/or C
8:     end while
9:     return results
10: end procedure
```

6.1.5 Complexity

We recall that the SVC is composed of two steps: the *cluster description* and the *cluster labeling*. To calculate the overall time and space complexity we have to analyze each step separately.

6.1.5.1 Cluster description complexity

The complexity of the cluster description is the complexity of the QP problem (see Equation 6.3) we have to solve for finding the MEB. Such a problem has $O(n^3)$ worst-case running time complexity and $O(n^2)$ space complexity. Anyway, the QP problem can be solved through efficient approximation algorithms like *Sequential Minimal Optmization (SMO)* (Platt, 1998) and many other decomposition methods. These methods can practically scale down the worst-case running time complexity to (approximately) $O(n^2)$, whereas the space complexity can be reduced to $O(1)$ (Ben-Hur et al., 2001, sec. 5).[12]

6.1.5.2 Cluster labeling complexity

The cluster labeling is composed of two sub-steps: (i) the construction of the adjacency matrix A (see Equation 6.19) and (ii) the computation of the connected components of the undirected graph induced by the matrix A. The size of such a matrix is $\overline{n} \times \overline{n}$, where $\overline{n} = n - n_{bsv}$.
In the first sub-step we have to compute the sphere radius R, i.e. the distance $d_R(\vec{s})$ where $\vec{s}$ is anyone of the support vectors, and the distance $d_R(\vec{y})$ for each point $\vec{y}$ sampled along the path connecting two points. The number m of points sampled along a path is the same for all paths we have to check. Finally, due to the dimensions of the adjacency matrix A, we have to check $\overline{n}^2$ paths.
Let us recall Equation 6.15

$$d_R^2(\vec{x}) = K(\vec{x},\vec{x}) - 2\sum_{k=1}^{n_{sv}} \beta_k K(\vec{x}_k,\vec{x}) + \sum_{k=1}^{n_{sv}}\sum_{l=1}^{n_{sv}} \beta_k\beta_l K(\vec{x}_k,\vec{x}_l)$$

[12] In the sequel we discuss the problem of the SVM training complexity more in details.

and let us consider the second *addendum* and the third one. The latter always assumes the same value for all distances computed among the points of the same dataset. Consequently, we can compute the third addendum only once and its cost is $O(n_{sv}^2)$. The computation of the second addendum costs $O(n_{sv})$.
Therefore, building the matrix A costs $O(n_{sv}^2) + O(\overline{n}^2 n_{sv} m)$. The first term can be absorbed into the second one, leading us to the $O(\overline{n}^2 n_{sv} m)$ as worst-case running time complexity for the matrix construction.
Once we have built the adjacency matrix A, we have to compute the connected components in order to obtain the final clustering. The algorithm for solving the connected components of an undirected graph costs $O(|V| + |E|)$ in time and $O(|V|)$ in space (Cormen et al., 2001), where $|V|$ is the number of vertices and $|E|$ is the number of edges. In our case $|V| = \overline{n}$ while $|E| \leq 2\overline{n} - 1$. So, the costs of the connected components algorithm can be absorbed into the costs needed for building the adjacency matrix.
The final worst-case running time complexity for the *cluster labeling* stage is $O(\overline{n}^2 n_{sv} m)$. The space complexity is $\Theta(\overline{n}^2) + \Theta(n)$, because the adjacency matrix A requires $\overline{n}^2$ memory units and the structure for the storage of the cluster labels requires only n memory units.

6.1.5.3 The overall complexity

The overall space complexity for a single iteration of the SVC is $O(\overline{n}^2)$, whereas the overall time complexity for a single iteration of the SVC is $O(n^2) + O(\overline{n}^2 n_{sv} m)$. Since the n_{bsv} is always much smaller than n, it results that $\overline{n}$ is only a little bit smaller than n. In addition, the term $\overline{n}^2$ is multiplied by n_{sv} and m. The n_{sv} term can even be equal to the 10% of n, while m is usually an integer value between 10 and 20. All these facts make the cluster labeling stage a bottleneck for the whole SVC algorithm.
Usually the number of iterations to converge is very small ($10 - 15$ iterations on average).

6.2 Issues and limitations

The original formulation of the SVC leaves several issues open and has some drawbacks and limitations.
Above all, we have seen that the cluster labeling algorithm complexity is a bottleneck for the whole clustering process. Besides, experimental results (Ben-Hur et al., 2001, sec. 4.1) showed that the SVC is not so robust with respect to dimensionality. Further investigation in this direction showed that the problem was in the *cluster labeling* stage. So, much effort was spent on the research for alternative cluster labeling algorithms (Lee and Lee, 2005, 2006; Lee and Daniels, 2005b, 2006; Nath and Shevade, 2006; Park et al., 2004; Yang et al., 2002).
Another big issue for the SVC is the tuning of the hyper-parameters: the identification of the right values for both the kernel width q and the soft margin constant C is crucial. As previously stated, these two parameters rule the whole clustering

process by determining the fraction of the SVs and BSVs, in order to detect the best clustering.
Without a robust kernel width exploration method, the SVC becomes an infeasible approach to tackle practical problems. Currently there are two methods for kernel width exploration. The first one is a robust secant-like algorithm (Lee and Daniels, 2004, 2005a,b), whereas the second one is a recently published angle-decrement algorithm inspired by the previous one (Hartono and Widyanto, 2007).
The estimation of soft margin constant C is still unexplored, although some embryonal ideas was proposed (Lee and Daniels, 2005a, sec. 3).
The third remarkable problem is the lack of a more accurate and robust stopping criterion. The threshold on the number of support vectors is not enough. Besides, such a stopping criterion introduces another issue: the estimation of such a threshold, which is clearly different according to the problem at hand.
Some minor problems remain. First, there was no further research on kernels, but it would be interesting to perform some experiments with kernel functions other than the Gaussian kernel. Finally, research to further reduce the overall time complexity would be very interesting too.
In the sequel we will present the last researches about the SVC and our own contributions. First, we provide a deeper investigation about algorithms for the cluster labeling stage and hyper-parameters tuning, together with some original contributions to the solution of this problem. Further contributions about alternative stopping criteria will be also provided. Moreover, we will study in depth which kernels could be suitable for the Support Vector Clustering. Then, we conclude the discussion about the SVC by presenting some works and some suggestion for further scaling down the overall computational and space complexities.

6.3 Cluster labeling algorithms

The original cluster labeling stage, referred to as the *Complete Graph (CG)* cluster labeling after its first publication (Ben-Hur et al., 2001), is a bottleneck for the whole clustering process. Another issue of that approach is the necessity of sampling a path between two points to check whether the line segment connecting two points exit the sphere in feature space. As already stated, the sampling involves a (obviously) limited number m of points along the path and this creates a *running time versus accuracy tradeoff*. If m is too high, the multiplicative factor introduced in the overall time complexity of the SVC becomes unacceptable. On the contrary, small values of m may cause two types of errors, i.e. False Negative (FN) and False Positive (FP) points (see Figure 6.6 and Figure 6.7). Furthermore, since the mapping into the feature space can be highly nonlinear, it is not clear if a line sampled in the data space is an appropriate means for making clustering decisions.
Therefore, most effort was spent for researching valid alternatives for this crucial step, in order to develop faster and robust algorithms keeping the quality of clustering results high. This section aims at reviewing all the remarkable works in this direction, which are evolved from simple modifications of the original procedure

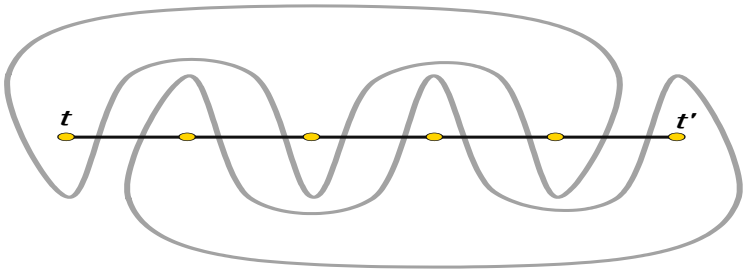


Figure 6.6: *A false positive situation: all sample points on $\overline{tt'}$ are inside the minimal hypersphere although the two points are in different clusters (Lee and Daniels, 2005b, p. 69).*

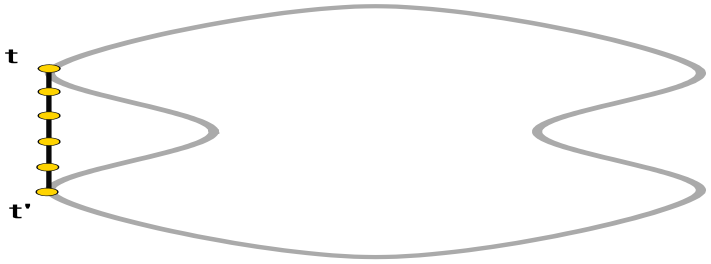


Figure 6.7: *A false negative situation: all sample points on $\overline{tt'}$ are outside the minimal hypersphere although the two points are in the same cluster (Lee and Daniels, 2005b, p. 69).*

towards totally new algorithms based on robust theories.

6.3.1 Support Vector Graph

(Ben-Hur et al., 2001) proposed a modification of the Complete Graph (CG) cluster labeling strategy, called *Support Vector Graph (SVG)*. It consists of a simple heuristics: the adjacency matrix A is not built by checking the linkages between all pair of data points anymore, but only considering the linkages between points and support vectors. The matrix A is now a $\overline{n} \times n_{sv}$ matrix, where $\overline{n} = n - n_{bsv}$. Due to these changes the undirected graph induced by the new matrix A is a subgraph of the one built with the CG procedure and the clusters are now identified by finding the connected components of this subgraph.

6.3.1.1 Complexity

The space complexity is $\Theta(\overline{n}n_{sv}) + \Theta(n)$. Consequently, we have to check only $\overline{n}n_{sv}$ paths. Therefore, building the matrix A costs $O(n_{sv}^2) + O(\overline{n}n_{sv}^2 m)$. The first term can be absorbed into the second one, resulting in $O(\overline{n}n_{sv}^2 m)$.
Once we have built the adjacency matrix A, we have to compute the connected components in order to obtain the cluster labeling for the support vectors. In this case the number of vertices is $|V| = \overline{n}$ while the number of edges is $|E| \leq 2n_{sv} - 1$. Again, the costs of connected components algorithm can be absorbed into the

Algorithm 6 Complete Graph Cluster Labeling

```
1: procedure CG(𝒳)
2:     for all x⃗ ∈ 𝒳 do                                  ▷ Build the matrix A
3:         for all y⃗ ∈ 𝒳 do
4:             if internalPath(x⃗, y⃗, m) then    ▷ check line segment using m points
5:                 A_{x⃗y⃗} = 1
6:             else
7:                 A_{x⃗y⃗} = 0
8:             end if
9:         end for
10:    end for
11:    clusterLabels ← findConnectedComponents(A)
12:    return clusterLabels
13: end procedure
```

costs needed for building the adjacency matrix, so the final worst-case running time complexity for the SVG cluster labeling procedure is $O(\overline{n}n_{sv}^2 m)$.
However, if n_{sv} is over $0.05n$ for $m = 20$ or $0.10n$ for $m = 10$ (these are common cases), the time complexity of this cluster labeling strategy become $O(\overline{n}n)$.

6.3.1.2 Conclusion

Although early experiments showed that this cluster labeling strategy is equivalent to the CG strategy, further tests have shown that the loss of information we find in the reduced graph causes less accurate clustering results. The observation is that the adjacencies between points are only kept within *neighborhoods* in the same cluster in the feature space. It is more likely that a point within a cluster has linkages to its neighbors, rather than to the boundary points of the cluster, i.e. the support vectors. Checking only adjacencies with support vectors may generate a number of unlinked data points, which turn into false negatives and make the clustering results less accurate. This situation can occur when the parameter C is too low, i.e. too many SVs become BSVs and there are only a few free SVs left.

6.3.2 Proximity Graph

The *Proximity Graph (PG)* cluster labeling strategy was the first real alternative to the originally proposed cluster labeling procedures. Yang et al. (2002) have performed several experiments with the Support Vector Clustering and have observed the already mentioned drawbacks of the original proposed cluster labeling techniques: CG tests many redundant edges, resulting in a highly computational demanding algorithm, whereas the SVG does not exactly decode neighborhood information, leading to trivial or meaningless clustering results.
The proximity graphs are a valid alternative to the measurement of the proximity by means of similarity instruments (e.g. Euclidean distance), especially if we have to work in very high dimensional spaces. In proximity graphs, vertices represent

Algorithm 7 Support Vector Graph Cluster Labeling

```
1: procedure SVG(𝒳)
2:     for all x⃗ ∈ 𝒳 do                                  ▷ Build the matrix A
3:         for all y⃗ ∈ 𝒳 : y⃗ is a support vector do
4:             if internalPath(x⃗, y⃗, m) then    ▷ check line segment using m points
5:                 A_{x⃗y⃗} = 1
6:             else
7:                 A_{x⃗y⃗} = 0
8:             end if
9:         end for
10:    end for
11:    ClusterLabels ← findConnectedComponents(A)
12:    return ClusterLabels
13: end procedure
```

data points and edges connect pairs of points to model proximity and adjacency. Despite the fact that the point is the most primitive object, it is no easy to define point proximity as a discrete relation. To best describe proximity between data points, the authors investigated and compared different graphs belonging to a common family of proximity graphs. These graphs are, for example, Delaunay Diagrams (DD), Minimum Spanning Trees (MST) and k-Nearest Neighbors (k-NN). By choosing appropriate underlying graphs the expected time complexity is sub-quadratic for data of all dimensions.

6.3.2.1 The strategy

The cluster labeling through proximity graph modeling constructs appropriate graphs to model a dataset. After the computation of the MEB, the obtained cut-off criterion, i.e. the sphere radius R, is adapted to estimate the edges of a proximity graph. This method avoids redundant checks in a complete graph and also avoids the loss of the neighborhood information. Under certain cut-off conditions, some points will become non-connected. Once again, the clusters are the connected components of the adjacency matrix A. The idea is to calculate coefficients of the adjacency matrix only for pairs $(\vec{x}_i, \vec{x}_j)$ such that $\vec{x}_i$ and $\vec{x}_j$ are linked by an edge E_{ij} in a proximity graph. The connectivity between the points is still decided by means of the path-sampling strategy already used in CG and SVG. Actually, the adjacency matrix A is not held explicitly in the memory, but it is encoded in the proximity graph. All edges in a proximity graph are called *candidate edges*. An edge E_{ij} is said to be an *active edge* if and only if the $A_{ij} = 1$; otherwise E_{ij} is a *passive edge*. A path is said to be an *active path* if and only if every edge in it is an active edge. So, a connected component in now equivalent to an active path.
Passive edges are not of interest and they will be removed from the proximity graph. After the removal of such edges, the task becomes recognizing all active paths formed. Once we have detected all active edges, we have all the information for collecting the connected components. As in the CG and SVG, the PG

Algorithm 8 Proximty Graph Cluster Labeling

```
1: procedure PG(𝒳)
2:     G ← buildProximityGraph(𝒳, R)    ▷ use the radius R as cut-off criterion
3:     G' ← removePassiveEdges(G)
4:     markActivePaths(G')
5:     ClusterLabels ← findConnectedComponents(G')
6:     return ClusterLabels
7: end procedure
```

cluster labeling strategy leaves BSVs unclassified. They are successively assigned to closest clusters, as already proposed in subsection 6.1.2.
In a proximity graph, points are connected by edges if they are close to each other according to some proximity measure. Near-by points are naturally more likely to be in the same cluster than points that are far away. Thus, cluster labeling via proximity graphs seems to be a good heuristics to reduce the time for testing linkages. In Yang et al. (2002) the previously cited proximity graphs have been investigated. They can be derived by considering different aspects of proximity and topology. DD represents a *is-neighbor* relation. The MST is based on the local closeness of points: it is a subgraph of DD and encodes less proximity information. k-NN is based on distance concepts.

6.3.2.2 Complexity

Let us analyze the space complexity first, in order to dispose the basis for the analysis of the time complexity. The number of vertices is the same for every proximity graph we choose. The factor that makes the difference is the number of edges. In the DD, such a number is linear in size n for a 2D space, but is quadratic in size n for a 3D space. Generally, the number of edges in the DD is $\Theta(n^{\lceil d/2 \rceil})$, where d is the dimensionality of the input space (Lee and Daniels, 2005b, sec. 2.1.5).[13] Other proximity graphs are linear in size n whatever data space dimensionality is. The number of edges in MST is $n - 1$, while for k-NN the upper bound is $O(kn)$. On the other hand, the field of the computational geometry has developed $O(n \log n)$ time algorithms to build DD, MST and k-NN in spaces of various dimensions (Yang et al., 2002). We recall that, in our case, for each iteration of the proximity graph building process we have to compute also a path sampling between the points that are to be connected. Such a sampling costs $O(n_{sv}m)$, leading the PG cluster labeling time complexity to $O(n \log n n_{sv} m)$. This makes proximity graph modeling scalable to high-dimensional data.

6.3.2.3 Conclusion

The experiments preformed by Yang et al. (2002) showed that the more robust proximity graphs are the DD and k-NN (with appropriate k), whereas the MST

[13] It is trivial that the employment of the DD is useful only when we work with two-dimensional data points, i.e. it is useless in most of the real-world problems.

resulted fragile. The more interesting results are about the k-NN that, with $k > 3$, captures satisfactory proximity information for cluster labeling in several situations, even in high BSV regime. In fact in such situations experiments showed that the proximity graph modeling does not produce unexpected false negatives as the SVG does.
Hence, this is a first step through a substantial enhancement of the Support Vector Clustering.

6.3.3 Spectral Graph Partitioning

Park et al. (2004) provide the first attempt to face the problem of the misclassifications due to the path-sampling, rather than a new cluster labeling procedure for the SVC. In few words, this approach can be potentially used on top of all cluster labeling strategies for support vector clustering. It provides a way to reduce the aforesaid misclassifications by combining several adjacency matrices (one for each SVC iteration) and applying *Spectral Graph Partitioning (SGP)* to the resulting matrix for finally cluster label computation.
This strategy relies on the intrinsic properties of the SVC. In fact, as stated in previous sections, we adopt a divisive execution scheme which yields different clustering results depending on the hyper-parameters. Hence, it can be considered that the q plays a role of yielding multi-resolutional results: small values of q produce high resolution contours while larger values produce low resolution contours. Several adjacency matrices are combined, selecting them to represent different resolutions, from coarse to fine. The method aims at improving accuracy of cluster assignment and avoiding to merge clusters that are supposed to be separated.

6.3.3.1 The strategy

The SGP strategy for cluster labeling consists of two steps:[14]

- the construction of all adjacency matrices needed
- the multi-resolution combination of such matrices for the final cluster labeling

As already stated, the first step can be performed using any of existing (or not existing yet) cluster labeling strategies. The second step is composed of other two sub-steps. Let Z be the set of all adjacency matrices built during the SVC execution. First, a new matrix $\hat{A}$ is built as the linear combination of the all matrices in Z (Park et al., 2004, sec. 3.1). Next, the final cluster labeling is obtained by applying the QR decomposition and the spectral relaxation principle to the matrix $\hat{A}$ (Park et al., 2004, sec. 3.2).

[14]Since this approach uses consolidated instruments with roots in linear algebra, we do not go into the mathematical details, but we provide an overview of the strategy proposed.

6.3.3.2 Conclusion

Since this approach is neither widespread not well documented, the only experiments we found in literature are the ones in Park et al. (2004). Anyway, these experiments highlight a peculiar characteristic of this approach: it makes the whole clustering process less sensitive to the hyper-parameters q and C, as it focuses on the construction of a number of different resolution adjacency matrices in order to create a new, more accurate, adjacency matrix. Furthermore, this approach promises to avoid several false positive and false negative classifications caused by sampling the points on the path connecting two points in feature space.
The main drawback of this approach is essentially the lack of documentation and experiments. We found it only in Park et al. (2004) and with few experiments attached. Furthermore, no accurate analysis of the additional computational effort is available. However, the approach seems to be quite interesting.

6.3.4 Gradient Descent

A new cluster labeling algorithm using topological properties of a trained kernel distance function was introduced in Lee and Lee (2005). This algorithm efficiently solves cluster labeling in high dimensional spaces by using a *Gradient Descent (GD)* strategy. It has been further extended in Lee and Lee (2006) for bypassing the problem of sampling the points.
The basic idea underlying the originally proposed GD strategy is to identify a subset of the input data points, called SEPs. These points are then used to build an adjacency matrix considerably smaller than the one built by means of the CG strategy. All the remaining points are successively assigned to the cluster on the basis of the SEPs labeling. This leads to a faster cluster labeling algorithm compared with the older ones.
Even though several experiments showed a good accuracy of this cluster labeling strategy, in the original formulation the construction of the adjacency matrix still relies on the path-sampling between SEPs to decide whether two points are adjacent or not.
In the next subsections we detailedly present the strategy with the extended results introduced in Lee and Lee (2006), which eliminate the necessity of sampling the paths.

6.3.4.1 Formulation of the dynamical system

Let $r(\cdot)$ be an alias for the function $d_R^2(\cdot)$ (see Equation 6.15) and let $\overline{r}$ be an alias for R^2. Let us consider the sublevel set of the distance function $r(\cdot)$

$$L_r(\overline{r}) \equiv \{\vec{x}\colon r(\vec{x}) \leq \overline{r}\}. \tag{6.28}$$

The sublevel set above can be decomposed in several disjoint subsets

$$L_r(\overline{r}) = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_p \tag{6.29}$$

The disjoint connected sets $\mathcal{C}_i$, $i = 1, 2, \cdots, p$, correspond to the p clusters identified by the SVC process.
In view of the equation above, the cluster structure can be analyzed by exploring a topological property (e.g. connectedness) of the sublevel set. To characterize the topological property of the sublevel set $L_r(\overline{r})$, a dynamical system associated with the trained kernel distance function $r(\cdot)$ was built. It can be shown that each connected component of $L_r(\overline{r})$ is exactly composed of dynamically invariant sets, the so-called basin level cells, of the built system. This decomposition facilitates the cluster labeling of each data point and extend the clusters to enlarged clustered domains that constitute the whole data space.
Let us consider a negative gradient dynamical system associated with the function $r(\cdot)$

$$\frac{d\vec{x}}{dt} = -\nabla r(\vec{x}) \tag{6.30}$$

The existence of a unique solution (or trajectory) $x(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^d$ for each initial condition $x(0)$ is guaranteed since the function $r(\cdot)$ is twice differentiable and the norm of ∇r is bounded.
Let us provide some basic definitions in order to continue the exposition.

Definition 6.2 (Equilibrium point) *Let $\vec{\overline{x}}$ be a state vector. The vector $\vec{\overline{x}}$ is an* **equilibrium point** *of the system in Equation 6.30 if and only if $\nabla r(\vec{\overline{x}}) = 0$.*

Definition 6.3 (Hyperbolic equilibrium point) *Let $\vec{\overline{x}}$ be an equilibrium point of the system in Equation 6.30. It is said to be* **hyperbolic** *if and only if the Hessian matrix of $r(\cdot)$ at $\vec{\overline{x}}$, denoted by $\nabla^2 r(\vec{\overline{x}})$, has no zero eigenvalues.*[15]

Definition 6.4 (Stable equilibrium point) *Let $\vec{\overline{x}}$ be a hyperbolic equilibrium point of the system in Equation 6.30. It is said to be a* **Stable Equilibrium Point (SEP)** *if and only if all the eigenvalues of its corresponding Hessian are positive.*

Definition 6.5 (Unstable equilibrium point) *Let $\vec{\overline{x}}$ be a hyperbolic equilibrium point of the system in Equation 6.30. It is said to be an* **Unstable Equilibrium Point (UEP)** *if and only if all the eigenvalues of its corresponding Hessian are negative.*

Definition 6.6 (Unstable manifold) *Let $\vec{\overline{x}}$ be an equilibrium point. We define the its* **unstable manifold** *as*

$$W^u(\vec{\overline{x}}) \equiv \{x(0) \in \mathbb{R}^d : \lim_{t \rightarrow -\infty} x(t) = \vec{\overline{x}}\} \tag{6.31}$$

Definition 6.7 (Saddle point) *Let $\vec{\overline{x}}$ be a hyperbolic equilibrium point of the system in Equation 6.30. It is said to be a* **saddle point** *if and only if it is neither an SEP nor a UEP.*

Definition 6.8 (Index-z saddle point) *Let $\vec{\overline{x}}$ be a saddle point of the system in Equation 6.30. It is said to be an* **index-z saddle point** *if and only if its Hessian has exactly z negative eigenvalues.*

[15]Note that all the eigenvalues of $\nabla^2 r(\vec{\overline{x}})$ are real since it is symmetric.

Definition 6.9 (Invariant set) *Let $T \subseteq \mathbb{R}^n$. The set T is called a* **positively (negatively) invariant set** *of the system in Equation 6.30 if and only if every trajectory of Equation 6.30 which start in T remains in T for all $t \geq 0$ ($t \leq 0$).*

The next theorem[16] states the complete stability and the positive invariance property of each cluster in Equation 6.28 under the process of Equation 6.30.

Theorem 6.1 *For a given trained kernel distance function $r(\cdot)$, suppose that for any $\vec{x} \in \mathbb{R}^d$, each connected component of the sublevel set $L_r(\hat{r}) = \{\vec{x} \colon r(\vec{x}) \leq \hat{r}\}$ is compact, where $\hat{r} = r(\vec{x}_0)$. Then, Equation 6.30 is completely stable, i.e. every trajectory of Equation 6.30 approaches one of the equilibrium points. Furthermore, each connected component of $L_r(\hat{r})$ is positively invariant, i.e. if a point is on a connected component of $L_r(\hat{r})$, then its entire positive trajectory lies on the same component.*

If we consider a Gaussian kernel, it can be shown that a trained kernel distance function $r(\cdot)$ satisfies the condition of the theorem above; that is, each connected component of the set $L_r(\hat{r}) = \{\vec{x} \colon r(\vec{x}) \leq \hat{r}\}$ is compact. Unless otherwise specified, from now on we will assume that the trained kernel distance function $r(\cdot)$ satisfies this condition.

6.3.4.2 Cluster decomposition

Let us introduce the *basin cell* and the *basin level cell* notions. They allow to decompose the whole data space into several separated clustered domains.

Definition 6.10 (Basin cell and basin level cell) *The* basin of attraction *of an SEP, $\vec{s}$, is defined as*

$$A(\vec{s}) \equiv \{x(0) \in \mathbb{R}^d \colon \lim_{t \to \infty} x(t) = \vec{s}\} \tag{6.32}$$

and the closure of the basin $A(\vec{s})$, denoted by $\overline{A(\vec{s})}$, is called a **basin cell**. *The boundary of the basin cell defines the* basin cell boundary, *denoted by $\partial\overline{A(\vec{s})}$.*
Moreover, the basin level set of $\vec{s}$, relative to a level value $\hat{r}$, is defined as the set of points in the sublevel set $L_r(\hat{r})$ that converges to the SEP $\vec{s}$ when Equation 6.30 is applied; that is

$$B_{\hat{r}}(\vec{s}) \equiv \{x(0) \in L_r(\hat{r}) \colon \lim_{t \to \infty} x(t) = \vec{s}\} \tag{6.33}$$

and the closure of the basin level set $B_{\hat{r}}$, denoted by $\overline{B_{\hat{r}}}$ is called a **basin level cell**.

From a topological viewpoint, $A(\vec{s})$ and $\overline{A(\vec{s})}$ are both connected and invariant. The following theorem shows that each cluster can be decomposed into the basin level cells of its constituent SEPs.

Theorem 6.2 *Let $S = \{\vec{s}_i\}_{i=1,2,\cdots,l}$ be the set of all SEPs of the Equation 6.30 in a nonempty sublevel set $L_r(\overline{r})$. Then, the following facts hold*

[16] A proof of this theorem is available in Lee and Lee (2005). On the contrary, the proofs of every other theorem concerning the GD strategy are available in Lee and Lee (2006).

- *Each basin level cell is given by*

$$\overline{B_{\overline{r}}(\vec{s}_i)} = \overline{A(\vec{s}_i)} \cap L_r(\overline{r}) \tag{6.34}$$

and is connected and positively invariant.

- *The sublevel set $L_r(\overline{r})$ is decomposed into the basin level cells*

$$L_r(\overline{r}) = \{\vec{x} \colon r(\vec{x}) \leq \overline{r}\} = \bigcup_{i=1}^{l} \overline{B_{\overline{r}}(\vec{s}_i)} \tag{6.35}$$

- *If $(\vec{s}_{i_j})_{j=1,2,\cdots,l_i}$ is the set of all SEPs in a cluster $\mathcal{C}_i$, then*

$$\mathcal{C}_i = \bigcup_{j=1}^{l_i} \overline{B_{\overline{r}}(\vec{s}_{i_j})} \tag{6.36}$$

The above theorem implies that, for any data point $\vec{x} \in L_r(\overline{r})$, there exists a corresponding SEP, say $\vec{s}_i$, such that $\vec{x} \in \overline{B_{\overline{r}}(\vec{s}_i)}$ and the cluster to which data point $\vec{x}$ belongs is identical to the cluster to which its related SEP $\vec{s}_i$ belongs too. Therefore, the *cluster labeling* of each data point can be accomplished by identifying the cluster label of its correlated SEP to which the data point converges by Equation 6.30.
Another important and unique consequence of Theorem 6.2 is the capability to extend the clusters obtained to enlarged clusters, as follows: since any point in the basin cell $\overline{A(\vec{s}_i)}$ enters into the basin level cell $\overline{B_{\overline{r}}(\vec{s}_i)}$ when Equation 6.30 is applied, $\overline{A(\vec{s}_i)}$ (which is connected and invariant) can be considered a natural extension of $\overline{B_{\overline{r}}(\vec{s}_i)}$. Therefore, if $(\vec{s}_{i_j})_{j=1,2,\cdots,l_i}$ is the set of all SEPs in a cluster $\mathcal{C}_i$, then such a cluster can be extended to an enlarged cluster $\mathcal{C}_i^E \supset \mathcal{C}_i$ given by

$$\mathcal{C}_i^E = \bigcup_{j=1}^{l_i} \overline{A(\vec{s}_{i_j})} \tag{6.37}$$

Since the basins $A(\vec{s}_i)$ are disjoint and the whole input space $\mathbb{R}^d$ is composed of basins cells $\overline{A(\vec{s}_{i_j})}$, the whole input space is partitioned into separated clustered domains, i.e.

$$\mathbb{R}^d = \mathcal{C}_1^E \cup \mathcal{C}_2^E \cup \cdots \cup \mathcal{C}_p^E \tag{6.38}$$

This extension is an outstanding property of this cluster labeling strategy: it is a natural way to assigns cluster labels to unknown data points.

6.3.4.3 Cluster structure characterization

Let us introduce now the concepts of an adjacency and a transition point. By means of these two concepts we can explore the topological structure of the clusters generated by the SVC and construct a weighted graph, simplifying the clusters structure. Such a graph provides a robust way to differentiate between the SEPs that belong to different clusters.

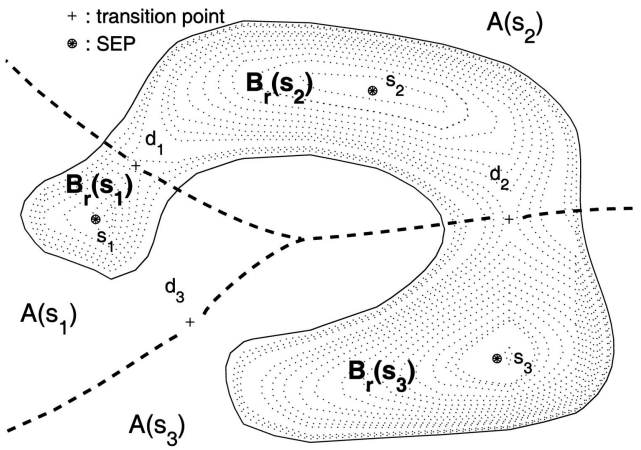


Figure 6.8: *The shaded regions represent clusters of $L_r(\overline{r})$ whose boundary is denoted by solid lines. The fulfilled dots are the SEPs, while the plus signs are the transition points (Lee and Lee, 2006, fig. 1a).*

Before introducing the notions of adjacency and transition point, it is necessary to provide a result showing a dynamic relationship between an SEP and an index-1 saddle point lying on its basin cell boundary.

Proposition 6.1 *Let $\vec{s}_1$ be an SEP of Equation 6.30. Then, there exists an index-1 saddle point $\vec{d} \in \partial\overline{A(\vec{s}_1)}$ such that the $1D$ unstable manifold (or curve) $W^u(\vec{d})$ converges to another SEP, say $\vec{s}_2$.*

From the proposition above we derive the following definition.

Definition 6.11 *Two SEPs, $\vec{s}_1$ and $\vec{s}_2$, are said to be* adjacent *to each other if there exists an index-1 saddle point $\vec{d} \in \overline{A(\vec{s}_1)} \cap \overline{A(\vec{s}_2)}$. Such an index-1 saddle point is called a transition point between the two SEPs.*

The existence of a transition point is necessary for the adjacency of two SEPs in addition to the condition that two basin cells intersect. The next theorem shows that the transition points are sufficient to decide whether two SEPs are in the same cluster.

Theorem 6.3 *Let $\vec{s}_i, \vec{s}_j$ be two adjacent SEPs. If $r(\vec{d}) < \overline{r}$ for a transition point $\vec{d}$ between $\vec{s}_i$ and $\vec{s}_j$, then such SEPs are in the same cluster of $L_r(\overline{r})$.*[17]

The concepts of adjacent SEPs and transition points allow us to build a weighted graph $G_{\overline{r}} = (V, E)$, describing the connections between SEPs, with the following elements

[17] The converse of this theorem is not generally true.

Algorithm 9 Find the SEPs for Gradient Descent Cluster Labeling

procedure FINDSEP($\mathcal{X}$)
  $M = 0$ ▷ the number of local minima
  $V = \emptyset$ ▷ V is a set of SEPs, i.e. the vertex set
  for all $\vec{x}_k \in \mathcal{X}$ **do**
    ▷ integrate Equation 6.30 starting from $\vec{x}_k$ until it reaches an SEP $\vec{x}_k^*$
    $\vec{x}_k^* \leftarrow$ integrateDynamicSysEq($\vec{x}_k$)
    if $\vec{x}_k^* \notin V$ **then** ▷ create $\langle \vec{s}_{M+1} \rangle$
      $\langle \vec{s}_{M+1} \rangle \leftarrow \vec{x}_k^*$
      $V \leftarrow V \cup \{\vec{s}_{M+1}\}$
      $\vec{x}_k \in \langle \vec{s}_{M+1} \rangle$
      $M \leftarrow M + 1$
    else
      find $\vec{s}_i \in V$ such that $\vec{x}_k^* = \vec{s}_i$
      $\vec{x}_k \in \langle \vec{s}_i \rangle$
    end if
  end for
  return M, V
end procedure

- The vertices in V are the p SEPs of Equation 6.30, with $r(\vec{s}_i) < \overline{r}$, $i = 1, 2, \cdots, p$
- The edge set E is defined as follows. The edge $[\vec{s}_i, \vec{s}_j] \in E$ with the weight $w[\vec{s}_i, \vec{s}_j] = r(\vec{d}_i)$ if there is a transition point $\vec{d}_i$ between $\vec{s}_i$ and $\vec{s}_j$ such that $r(\vec{d}_i) < \overline{r}$.[18]

The constructed graph simplifies the clusters structure of the sublevel set $L_r(\overline{r})$ and gives us an insight into the topological structures of the clusters. The next theorem establishes the equivalence of the topological structures between the graph $G_{\overline{r}}$ and the clusters of the sublevel set $L_r(\overline{r})$.

Theorem 6.4 *Each connected component of the graph $G_{\overline{r}}$ corresponds to a cluster of the sublevel set $L_r(\overline{r})$. That is, $\vec{s}_i$ and $\vec{s}_j$ are in the same connected component of the graph $G_{\overline{r}}$ if and only if $\vec{s}_i$ and $\vec{s}_j$ are in the same cluster of the sublevel set $L_r(\overline{r})$.*

This theorem enables us to distinguish SEPs that belong to different clusters by constructing the weighted graph $G_{\overline{r}}$. Finding the connected components of such a graph leads us to the labeling of the SEPs; succesively, we can label other points assigning them the label of the corresponding SEP.
Algorithm 11 shows that for finding the graph we need to: (i) find the SEPs (so the vertices), (ii) find all equilibrium points of the system, (iii) calculate the transition points (to build the edges) (Lee and Lee, 2006, sec. 5).
The computation of transition points is crucial in constructing a graph whose topological structure is equivalent to that of the clusters described by a trained

[18]Note that the edge weights always take positive values from Equation 6.15.

Algorithm 10 Gradient Descent Cluster Labeling - Original Version

```
1: procedure GD(𝒳)
2:     (M, V) ←findSEP(𝒳)
3:     for i ← 1, M do
4:         for j ← 1, M do
5:             if internalPath(s⃗_i, s⃗_j, m) then   ▷ check line segment using m points
6:                 A_ij = 1
7:             else
8:                 A_ij = 0
9:             end if
10:        end for
11:    end for
12:    ClusterLabels ← findConnectedComponents(A)          ▷ labeling SEPs
13:    for i ← 1, M do                                     ▷ labeling non-SEP points
14:        for all w⃗ ∈ {s⃗_i} do
15:            ClusterLabels[w⃗] ← getClusterLabel(s_i)
16:        end for
17:    end for
18:    return ClusterLabels
19: end procedure
```

kernel distance function. Moreover, it helps to correctly assign a cluster label to each SEP from the vertices consisting of only SEPs. This is the main and substantial enhancement provided in Lee and Lee (2006). In fact, in Lee and Lee (2005) the widely used CG labeling strategy was applied to build an adjacency matrix consisting of only SEPs (see Algorithm 10), but using a sampled path to decide the adjacencies may produce several misclassifications.

6.3.4.4 Complexity

Algorithm 10 and Algorithm 11 show the pseudocode for the two versions of the GD labeling strategy. Algorithm 9 shows the pseudocode to build the disjoint groups represented by an SEP each (see Equation 6.29) and it is shared between the two versions of the GD labeling strategy.
The worst case running-time of the Algorithm 9 is $O(n^2t)$. The term t represents the number of iterations necessary for the numerical integration to converge to an SEP.
In the Algorithm 10 the construction of the adjacency matrix takes $O(M^2n_{sv}m)$, where M is the number of the SEPs found with Algorithm 9 and $n_{sv}m$ is the already mentioned cost for sampling a path. Next, the cost of finding the connected components can be absorbed in the cost of building the adjacency matrix. Finally, we have the labeling of the remaining (non-SEP) points, which costs $O(n - M)$. Therefore, the overall worst case running-time complexity for the Algorithm 10 is $O(n^2(t + nv_{sv}m))$.
The space complexity for the Algorithm 10 is $O(M^2)$ for the storage of the adja-

Algorithm 11 Gradient Descent Cluster Labeling - Enhanced Version

```
1: procedure GDE(𝒳)
2:     (M, V) ←findSEP(𝒳)
3:     EP ← findDynamicSysEquilibriumPoints()    ▷ find EPs of Equation 6.30
4:     E ← findTransitionPoints(EP, H)    ▷ H is the Hessian ∇²r(·), E the edge set
5:     G_r̄ ← createWeightedGraph(V, E)
6:     ClusterLabels ← findConnectedComponents(G_r̄)
7:     for i ← 1, M do                                   ▷ labeling non-SEP points
8:         for all w⃗ ∈ {s⃗_i} do
9:             ClusterLabels[w⃗] ← getClusterLabel(s_i)
10:        end for
11:    end for
12:    return ClusterLabels
13: end procedure
```

cency matrix and $O(n)$ for the storage of cluster labels. Since for large values of n we usually have $M \ll n$, the overall space complexity for this algorithm is $O(n)$.
Let us analyze now the Algorithm 11. We have not an adjacency matrix anymore, but we finish the construction of the weighted graph by computing all equilibrium points and (then) the transition points of the system. The computation of the equilibrium points is linear in n with some hidden constants (see Lee and Lee (2006, sec. 5)), whereas the computation of the transition points is linear in the number q of the equilibrium points found. The computation of the transition points also require the numerical integration and so we have other hidden constants. Anyway, the worst case running time complexity still is quadratic, $O(n^2 t)$, without taking into account some hidden constants.
Experimental tests reveal that the execution time of the enhanced algorithm is about twice the execution time of the original formulation of the algorithm. However, for large values of n the execution time of the two versions becomes similar and similar (Lee and Lee, 2006, table 1).
The space complexity does not change in the enhanced version of the GD strategy.

6.3.4.5 Conclusion

The Gradient Descent (GD) cluster labeling strategy is the first complex strategy based on robust theoretical fundamentals. Several experiments have shown a great accuracy in cluster labeling. The main advantages introduced with this strategy are the elimination of the path sampling and the identification of the cluster labels on the basis of those points (SEPs) which lie in the high density probability region of the corresponding clusters. Furthermore, a unique characteristic of this approach is the ability of extend clusters to enlarged clustered domains which partition the whole data space $\mathbb{R}^d$ in p disjoint subdomains, where p is the number of clusters. This enable us to decide in what cluster completely unknown points have to be put. Finally, the theory behind the GD strategy also

suggests a robustness with respect to the high dimensionality.
The main drawback is the computational complexity: the worst case running-time still results quadratic in size n. Anyway, experimental results have shown that the average computational complexity of this strategy is substantially lower than the average computational complexity of the previously presented strategies.

6.3.5 Cone Cluster Labeling

Cone Cluster Labeling (CCL) is another recent cluster labeling strategy (Lee and Daniels, 2005b, 2006). It was pretty much developed in the same period the first version of Gradient Descent (GD) labeling was. The CCL is the second labeling strategy for SVC which relies on robust theoretical fundamentals. CCL is the faster (but not necessarily the more accurate) cluster labeling algorithm available for the SVC and it is able to work with very high dimensional data. Moreover, CCL also eliminates the problem of sampling a path to decide about the adjacency of two points.
The main idea of this approach is to try to cover a key portion of the minimal hypersphere in feature space by means of cones that are anchored at each support vector in feature space. It can be shown that such cones in feature space correspond to hyperspheres in data space and the union of such hyperspheres forms an approximate covering of the data space contours. The union needs not to be explicitly constructed. Pairs of support vectors are quickly tested during the labeling process and then the remaining points can be easily clustered.

6.3.5.1 Overview

Strictly speaking, the approximate covering is not for the MEB in feature space, but for the intersection P in feature space between the surface of the unit ball and the MEB.[19] Such an intersection is shaped like a cap. When P is mapped back to the data space, we obtain the contours of $\phi^{-1}(P) \cap \mathbb{R}^d = P'$. The approximate covering consists of a union of cone-shaped regions. Each region is associated with the feature space image of a support vector. Let $V = \{\vec{v}_i : 0 < \beta_i < C\}_{i=1,2,\cdots,n_{sv}}$ be the set of SVs for a given kernel width value. The region for support vector $\vec{v}_i$ is denoted by $E_{\vec{v}_i}$ and is called a *support vector cone*.
Let B be the surface of the feature space unit ball. A hypersphere $S_{\vec{v}_i}$ centered on $\vec{v}_i$ in the data space maps to a subset of $E_{\vec{v}_i} \cap B$ and is called a *support vector sphere*. In the sequel it will be shown that the radii of these support vector spheres are all the same; this contributes to the speed of the CCL.
The union of the support vector spheres

[19] The original formulation of CCL assumes the employment of the Gaussian kernel. We will show in the sequel that the Gaussian kernel is a normalized kernel, i.e. $\forall \vec{x} \in \mathcal{X}, K(\vec{x}, \vec{x}) = 1$. This implies that the feature space hypersphere is never larger than the unit ball. For more generality, we can assume a generic normalized kernel (see subsection 6.7.2).

$$\bigcup_{i=1}^{n_{sv}} S_{\vec{v}_i} \tag{6.39}$$

is an approximate covering of the data space contours P'. However, the union is not explicitly constructed. Rather, cluster labeling is performed in two phases. First, support vectors are clustered; two SVs, $\vec{v}_i$ and $\vec{v}_j$ are supposed to be connected if their support vector spheres overlap, $S_{\vec{v}_i} \cap S_{\vec{v}_j} \neq \emptyset$. Forming the transitive closure of the connected relation yields a set of support vector clusters. The second (and final) step is the clustering of the remaining points, that are assigned to the clusters of the support vectors they are closest to.

6.3.5.2 Covering of the feature space minimum enclosing ball

Both the covering of the sphere in the feature space and the related covering of contours in the data space use just the support vectors. Therefore, support vectors play a crucial role for the CCL.
Let $\Theta_i = \angle(\vec{a}O\phi(\vec{v}_i))$, where O is the feature space origin and $\vec{a}$ is the center of the MEB. In feature space, each support vector $\vec{v}_i$ has its own cone $E_{\vec{v}_i}$ that covers a portion of P. The support vector cone $E_{\vec{v}_i}$ is defined as the infinite cone with axis $\overrightarrow{\phi(\vec{v}_i)}$ and base angle Θ_i.

Lemma 6.2 $\forall \vec{v}_i, \vec{v}_j \in V, \angle(\phi(\vec{v}_i)O\vec{a}) = \angle(\phi(\vec{v}_j)O\vec{a})$

The above lemma shows[20] that the base angle is the same for all support vector cones. Therefore, we can denote all base angles simply by Θ.
To approximately cover P, the center of the MEB is projected onto the surface of the unit ball. The projected point is denoted by $\vec{a'}$ (see Figure 6.9 (b)). The point $\vec{a'}$ is a common point of intersection for all support vector cones. Thus,

$$\bigcup_{i=1}^{n_{sv}} E_{\vec{v}_i} \cap P \approx P \tag{6.40}$$

Let us note that since $\vec{a'}$ is the projection of $\vec{a}$, by replacing $\vec{a}$ with $\vec{a'}$ in Lemma 6.2 we obtain the same conclusions, i.e.

$$\forall \vec{v}_i, \vec{v}_j \in V, \angle(\phi(\vec{v}_i)O\vec{a'}) = \angle(\phi(\vec{v}_j)O\vec{a'}) \tag{6.41}$$

6.3.5.3 Covering of data space contours

Let us start with a consideration: all data whose feature space images have a base angle smaller than Θ with respect to $\overrightarrow{\phi(\vec{v}_i)}$ are in the same cluster as the support vector $\vec{v}_i$. The goal in this section is to highlight the connection between points in data space and their images in feature space, in such a manner that we obtain a

[20]The proof of all theorems and lemmas concerning CCL can be found in Lee and Daniels (2005b).

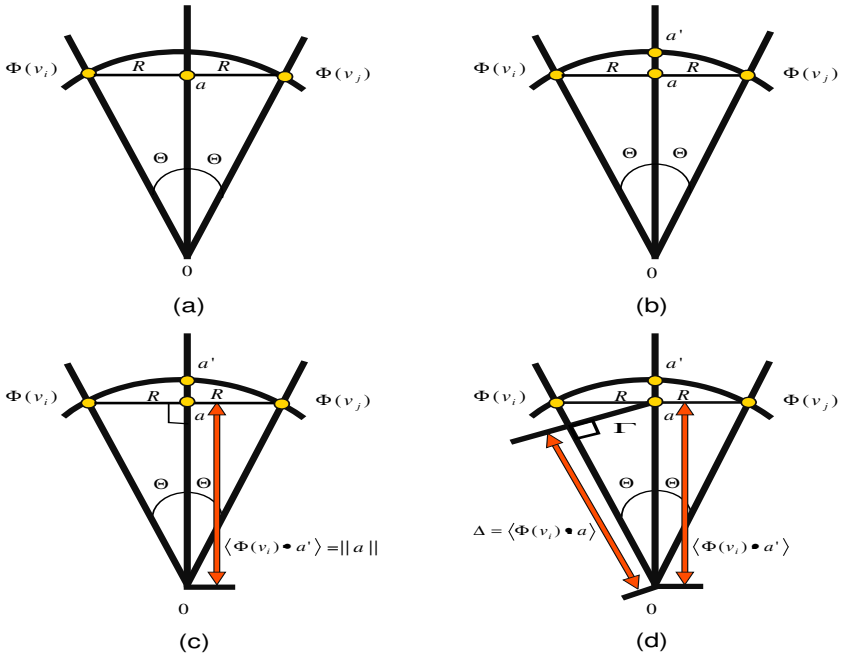


Figure 6.9: *Developing an approximate cover of a part of P, where $\vec{v}_i$ and $\vec{v}_j$ are SVs and $\phi(\vec{v}_i)$ and $\phi(\vec{v}_j)$ are their feature space images, respectively. (a) $\Theta = \angle(\phi(\vec{v}_i)O\vec{a}) = \angle(\phi(\vec{v}_j)O\vec{a})$; (b) $\vec{a'}$ is the intersection of $\vec{a}$ with the unit ball surface; (c) the length $\|\vec{a}\|$ is $\langle\phi(\vec{v}_i), \vec{a'}\rangle$; (d) $\Delta = \langle\phi(\vec{v}_i), \vec{a}\rangle = 1 - R^2 = \|\vec{a}\|^2$. Note that this is only a 2D illustration; the actual feature space is high-dimensional (Lee and Daniels, 2006, fig. 2).*

data space criterion to decide the cluster membership. Such a criterion employes data-space distances corresponding to feature-space angles.
Let $\vec{g} \in \mathbb{R}^d$ be a point such that $\phi(\vec{g}) = \vec{a'}$. All data points which have a distance from $\vec{v}_i$ smaller than $\|\vec{v}_i - \vec{g}\|$ correspond to feature space images that have an angle smaller than Θ with respect to $\overrightarrow{\phi(\vec{v}_i)}$ (in view of Equation 6.41).
To approximately cover P' in data space using support vector cones, the intersection $E_{\vec{v}_i} \cap P$ is mapped back to data space, obtaining a support vector sphere $S_{\vec{v}_i}$. Since all support vector cones have the same base angle Θ, all support vector spheres have the same radius Z and each of them is centered in a support vector. It can be shown that the radius Z is $\|\vec{v}_i - \vec{g}\|$.

Lemma 6.3 *Each data point $\vec{x} \in \mathcal{X}$ whose image $\phi(\vec{x})$ is inside $E_{\vec{v}_i} \cap P$, $\phi(\vec{x}) \in E_{\vec{v}_i} \cap P$, have a distance $D_{\vec{x}} = \|\vec{v}_i - \vec{x}\|$ such that $D_{\vec{x}} \leq \|\vec{v}_i - \vec{g}\|$.*

The claim implies that $E_{\vec{v}_i} \cap P$ corresponds to a hypersphere $S_{\vec{v}_i}$ in the data space centered at $\vec{v}_i$ with radius $Z = \|\vec{v}_i - \vec{g}\|$. Since the union of support vector cones approximately covers P (see Equation 6.40), the union of support vector spheres approximately covers P'

$$\bigcup_{i=1}^{n_{sv}} S_{\vec{v}_i} \cap P' \approx P' \tag{6.42}$$

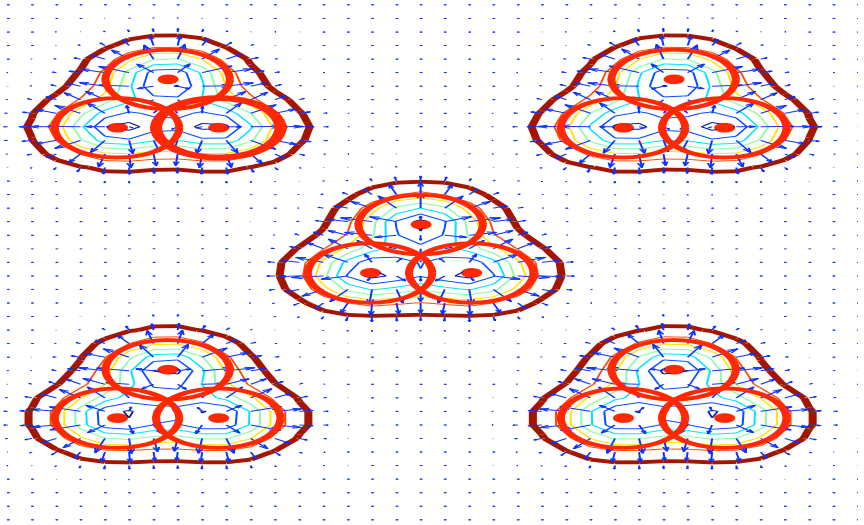


Figure 6.10: *Support vector cones projections into data space, i.e. support vector spheres. They are represented by circles with the same radius. The outermost contours are the real contours of the clusters (Lee and Daniels, 2005b, fig.16).*

The next task is to determine the radius $Z = \|\vec{v}_i - \vec{g}\|$. Because $\|\phi(\vec{v}_i)\| = 1 = \|\vec{a'}\|$, we have

$$\cos(\Theta) = \cos(\angle(\phi(\vec{v}_i)O\phi(\vec{g}))) = \cos(\angle(\phi(\vec{v}_i)O\vec{a'})) = \langle\phi(\vec{v}_i), \vec{a'}\rangle \tag{6.43}$$

Thus, we obtain[21]

$$Z = \|\vec{v}_i - \vec{g}\| = \sqrt{-\frac{\log\cos\Theta}{q}} \tag{6.44}$$

Note that since Θ is the same for all $\vec{v}_i \in V$, all $S_{\vec{v}_i}$ have the same radius.
In order to calculate $\cos\Theta$ we need to know the value of $\langle\phi(\vec{v}_i), \vec{a'}\rangle$. Following lemmas enable us to do this.

Lemma 6.4 $\forall\vec{v}_i \in V, \langle\phi(\vec{v}_i), \vec{a'}\rangle = \|\vec{a}\|$ *(see Figure 6.9 (c)).*

Corollary 6.1 $\overrightarrow{\phi(\vec{v}_i)\vec{a}}$ *is orthogonal to* $\vec{a}$

Lemma 6.5 $\forall\vec{v}_i \in V, \langle\phi(\vec{v}_i)\rangle, \vec{a} = 1 - R^2 = \|\vec{a}\|^2$ *(see Figure 6.9 (d)).*

From above results, we have

$$Z = \sqrt{-\frac{\log(\sqrt{1-R^2})}{q}} \tag{6.45}$$

[21] We recall that log refers to the natural logarithm.

6.3.5.4 Cluster labels assignment

At this point, we have all the instruments to formalize the cluster labels assignment. We first provide some basic definitions in order to properly define the connectivity among support vectors.

Definition 6.12 (Diameter Reachable) *Let $\vec{x}, \vec{y} \in \mathbb{R}^d$ be two points and let $z \in \mathbb{R}$ a radius value. The two points are said to be* **diameter reachable** *to each other with respect to z if and only if $\|\vec{x} - \vec{y}\| \leq 2z$. We denote that as $\vec{x} \oslash_{r,z} \vec{y}$.*

The *diamater reachable* relation is a transitive relation.

Definition 6.13 (Diameter Connected) *Let $\vec{x}, \vec{y} \in \mathbb{R}^d$ be two points and let $z \in \mathbb{R}$ a radius value. The two points are said to be* **diameter connected** *to each other with respect to z if and only if there exists a chain of data points $\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_t$ (with $\vec{x}_1 = \vec{x}$ and $\vec{x}_t = \vec{y}$) such that*

$$\forall i = 1, 2, \cdots, t-1, \vec{x}_i \oslash_{r,z} \vec{x}_{i+1}$$

We denote that as $\vec{x} \oslash_{c,z} \vec{y}$.

The *diameter connected* relation is the transitive closure of the *diameter reachable* relation.

Definition 6.14 (Cone Cluster) *Let $\mathcal{C} \subseteq \mathcal{X}$, $\mathcal{C} \neq \emptyset$, be a cluster and let $z \in \mathbb{R}$ a radius value. We call $\mathcal{C}$ a* **cone cluster** *if it satisfies the following conditions*

1. Maximality. $\forall \vec{x}, \vec{y} \in \mathcal{X}, \vec{x} \in \mathcal{C} \wedge \vec{x} \oslash_{r,z} \vec{y} \implies \vec{y} \in \mathcal{C}$
2. Connectivity. $\forall \vec{x}, \vec{y} \in \mathcal{C}, \vec{x} \oslash_{c,z} \vec{y}$

A *cone cluster* is simply a cluster defined on the basis of the CCL properties.

By means of these definitions, we can build the support vector adjacency matrix A. We have previously stated that two support vectors are said to be connected if their support vector hyperspheres overlap. Since all support vector spheres have the same radius and in view of diameter reachable definition, we can state that two support vectors are connected if and only if they are diameter reachable to each other with respect to the radius Z. Therefore, the adjacency matrix represents diameter connected points.
Once we have built the adjacency matrix A, we find the connected components of the graph induced by A in order to assign labels to the support vectors.
The final step is to assign the labels to remaining points on the basis of the labels assigned to support vectors. The natural way is to use Lemma 6.3 and assign a point to the cluster of the support vector it is closest to, such that the distance between the point and the support vector is less than or equal to Z. Unfortunately this strategy leads to leave some points unlabeled, so the constraint on the distance between the point and the support vector has been relaxed, by finding just the closest support vector. This strategy generally leads to very good results.

Algorithm 12 Cone Cluster Labeling

procedure CCL($\mathcal{X}, V, q, R$)
  $Z \leftarrow \sqrt{-\ln(\sqrt{1-R^2})/q}$
  for all $\vec{x} \in V$ **do** ▷ Build the matrix A
    for all $\vec{y} \in V$ **do**
      if $\vec{x} \oslash_{c,Z} \vec{y}$ **then**
        $A_{\vec{x}\vec{y}} = 1$
      else
        $A_{\vec{x}\vec{y}} = 0$
      end if
    end for
  end for
  clusterLabels $\leftarrow$ findConnectedComponents(A)
  for all $\vec{x} \in \mathcal{X} : \vec{x} \notin V$ **do**
    $\vec{v} \leftarrow$ findClosestSV($\vec{x}$)
    clusterLabels[$\vec{x}$] $\leftarrow$ getClusterLabel($\vec{v}$)
  end for
  return clusterLabels
end procedure

6.3.5.5 Complexity

To compute Z we need first to calculate the radius R of the MEB. By means of Equation 6.15 we can calculate R^2 directly and this costs $O(n_{sv}^2)$. The construction of the adjacency matrix also have $O(n_{sv}^2)$ time complexity and as usual the complexity of the procedure for finding the connected components is absorbed into the complexity for the construction of the adjacencies. Finally, the last step for labeling the non-SV points have $O((n - n_{sv})n_{sv})$ time complexity. Therefore, the overall time complexity of the CCL is $O((n - n_{sv})n_{sv})$, which makes CCL the fastest cluster labeling algorithm currently available for the SVC.

6.3.5.6 Conclusion and contribution

The CCL is the fastest labeling algorithm for the SVC and from experimental results we know also that it produces very high quality clustering results even in high dimensional spaces.
However the CCL is heavily based on approximation strategies which rely on support vectors. This may lead the labeling strategy to produce several misclassifications with complex dataset, especially datasets with a high level of noise: in this case the support vectors increasingly become less reliable, because we may use a high BSV regime to handle the noise. Moreover, with particular datasets, CCL may need very large kernel width values for producing meaningful labeling. The reason is that support vector cones are highly overlapped with the initial kernel width value and the cones shrink very slowly as the kernel width value is increased.

Algorithm 13 Cone Cluster Labeling - Version with our own contribution

procedure CCL2($\mathcal{X}, V, BSV, q, R$)
  $Z \leftarrow \sqrt{-\ln(\sqrt{1-R^2})/q}$
  for all $\vec{x} \in V$ **do** ▷ Build the matrix A
    for all $\vec{y} \in V$ **do**
      if $\vec{x} \oslash_{c,Z} \vec{y}$ **then**
        $A_{\vec{x}\vec{y}} = 1$
      else
        $A_{\vec{x}\vec{y}} = 0$
      end if
    end for
  end for
  clusterLabels $\leftarrow$ findConnectedComponents(A)
  for all $\vec{x} \in \mathcal{X} : \vec{x} \notin V \cup BSV$ **do**
    $\vec{v} \leftarrow$ findClosestSV($\vec{x}$)
    clusterLabels[$\vec{x}$] $\leftarrow$ getClusterLabel($\vec{v}$)
  end for
  for all $\vec{b} \in BSV$ **do**
    $\vec{v} \leftarrow$ findClosestPoint($\vec{b}$)
    clusterLabels[$\vec{b}$] $\leftarrow$ getClusterLabel($\vec{v}$)
  end for
  return clusterLabels
end procedure

We have also made a simple but effective modification to the CCL which provides better results in several cases (see Algorithm 13). We have modified the last step of the algorithm: instead of assigning labels to all non-SV points, we exclude BSV points from this step. Next, we assign labels to the BSVs putting them into the cluster of the point their are closest to. This does not change the computational complexity and it is very useful especially when we deal with problems that require a high BSV regime.

6.3.6 Other works on Cluster Labeling

There are other minor works on cluster labeling. The first one is a further modification of the SVG strategy, proposed by Nath and Shevade (2006). We recall SVG labeling constructs the adjacency matrix by checking the connectivity only between the whole set of points and the set of support vectors, leading to a matrix $(n - n_{bsv}) \times n_{sv}$ in size. The aforementioned minor modification further reduce the number of points used and relies only on SVs. However, this accentuates the problem of the weak connectivity due to the lack of the information brought by the internal points.
Another minor cluster labeling algorithm was presented as a combination of the SVG strategy and the PG one (with DD proximity graph) (Lee and Daniels, 2005b, 5.2). It relies on Delaunay triangulation of the support vectors. Despite its speed,

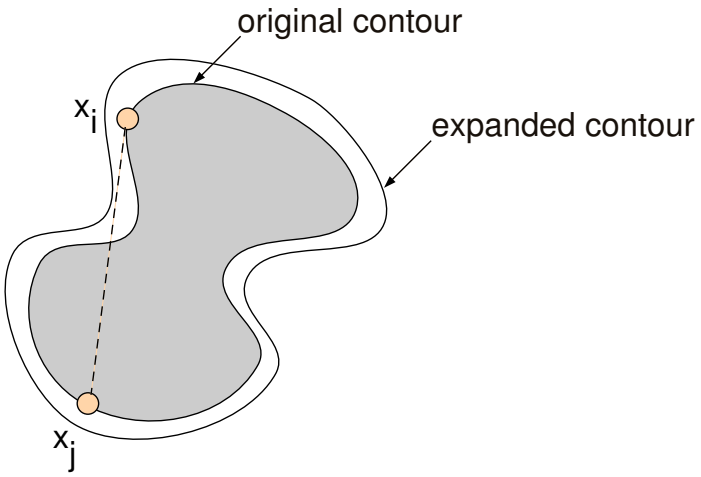


Figure 6.11: *Contour ϵ-expansion to face the weak connectivity. With the original contour the two points are detected as disconnected with the path sampling technique. The same two points result connected if we ε-expand the contour (Lee and Daniels, 2005b, fig.14).*

this cluster labeling is useless, because it works well only in two-dimensional spaces without outliers.

Finally a technique to face the connectivity weakness due to the sampling of points along a path was proposed (Lee and Daniels, 2005b, sec. 5.3): the *Minimal Sphere Radius Expansion*, which results also in contours expansion in data space. The radius of the found MEB is expanded by an ϵ in order to have expanded contours in data space (see Figure 6.11). This provides a way to reduce the weakness of the connectivity. Two points may result disconnected if the line segment connecting them is sampled within the original contours, even if they are really connected. The ϵ-expanded contour may bypass this problem in some cases.

6.3.7 Overall conclusion

The originally proposed cluster labeling strategies (CG and SVG) have been outperformed by several newer strategies. The PG strategy have a low computational complexity but still relies on the sampling of a path to decide the connectivity of two points. The more interesting ones are the GD and CCL strategies. Theoretically speaking, the enhanced version of the GD seems to be the most robust labeling strategy for the SVC. On the other hand, the optimal tradeoff between accuracy and speed of the CCL is very attractive, especially for very large datasets.

6.4 Kernel width exploration

Let us go deeper into the hyper-parameters tuning issue. We first present the kernel width exploration methods available for the SVC (Hartono and Widyanto, 2007; Lee and Daniels, 2005a). These exploration strategies rely on intrinsic properties of the SVC, so they result in *ad hoc* strategies.
The role of a kernel width exploration method is crucial. In the divisive execution scheme we perform the clustering process many times at different q values. Therefore, we aim at executing the SVC only for "useful" kernel width values. A brute force generation of the kernel width values is absolutely unfeasible, because the SVC would perform too much iterations and most of them would be redundant.
The secant-like algorithm for kernel width exploration was presented as a *Gaussian kernel width exploration* (Lee and Daniels, 2004, 2005a,b). In reality, we will show that the process is suitable for all normalized kernels, keeping open the chance of employing kernels other than the Gaussian one.
Since the angle-decrement exploration algorithm (Hartono and Widyanto, 2007) is inspired by the secant-like one, such a generalization still holds.

6.4.1 Secant-like kernel width generator

This method is based on the intuition that significant changes in clustering structure are less likely to occur in q intervals where R^2 values are fairly stable. The secant-like algorithm relies on R^2 monotonicity and the sequence of kernel width values is generated by creating a convex approximation to the graph of R^2 versus q. The estimate relies on spatial characteristics of the dataset and does not depend on the number of data points or the dataset dimensionality. Moreover, it is possible to estimate how many kernel width values can be generated for a given dataset, by using well-known results about secant methods convergence.
The theoretical fundamentals presented in the following subsections provide conditions under which R^2 is a monotonically nondecreasing function of q. The results hold for spaces of arbitrary dimensions. We only need to assume that the number of outliers is not varied during the process and the kernel used is normalized.[22]

6.4.1.1 R^2 as a function of q

Like in the previous sections, the proofs of all theorems and lemmas are omitted and can be found in Lee and Daniels (2005b, ch. 4).

Lemma 6.6 $\frac{1}{n} < C \leq 1$

To characterize R^2 as a function of q, we first observe that, as R is the radius of the MEB, $R^2 \geq 0$ for $0 \leq q \leq +\infty$. Moreover, the lemma below shows a stronger result.

[22] We recall that the Gaussian kernel satisfy this requirement. See section 6.7 for major details.

Lemma 6.7 $q = 0 \iff R^2 = 0$

Since we assume a normalized kernel,[23] i.e. a kernel $K(\cdot)$ such that $K(\vec{x}, \vec{x}) = 1$ for all $\vec{x} \in \mathcal{X}$, we have that the entire data space is embedded onto the surface of the unit ball in feature space, i.e. the MEB is never larger of the unit ball in feature space. Therefore, $0 \leq R^2 \leq 1$ for $0 \leq q \leq +\infty$. The following results lead to the stronger result that $R^2 \leq 1 - 1/n$.

Lemma 6.8 $q = +\infty \implies \forall i = 1, 2, \cdots n \ \beta_i = \frac{1}{n}$

Lemma 6.9 $q = +\infty \implies \forall i = 1, 2, \cdots n \ d_R^2(\vec{x}_i) = 1 - \frac{1}{n}$

Lemma 6.10 $R^2 = 1 \iff q = +\infty \wedge n = +\infty$

Corollary 6.2 *If n is finite, then $R^2 \leq 1 - \frac{1}{n} < 1$. Furthermore, $R^2 = 1 - \frac{1}{n}$ if and only if $q = +\infty$*

The results above combined with the ones below show that, under certain conditions, R^2 is a monotonically nondecreasing function of q. In the following results the q subscript refers to a specific kernel width value q.

Lemma 6.11 $\forall \vec{x}_i, \vec{x}_j \in \mathcal{X} \ q^{'} > q \implies \|\phi(\vec{x}_i) - \phi(\vec{x}_j)\|_{q^{'}} > \|\phi(\vec{x}_i) - \phi(\vec{x}_j)\|_q$

Now, let $S_q(\mathcal{X})$ be the minimal sphere for the set $\mathcal{X}$ and the kernel q. It follows from the Welzl algorithm (Welzl, 1991) that $S_q(\mathcal{X})$ in an s-dimensional space is determined by a set V of points (support vectors), such that $S_q(V) = S_q(\mathcal{X})$, whose images lie on the surface of $S_q(\mathcal{X})$, where $2 \leq |V| \leq \min(n, s + 1)$. V is said to be minimal if there is no $V^{'}$ such that $V^{'} \subseteq V \implies S_q(V^{'}) = S_q(V)$.

Theorem 6.5 *For $q^{'} > q$ and $C = 1$ in an s-dimensional feature space, suppose that there exists a set V of minimal support vectors for $q^{'}$ and q such that $S_{q^{'}}(V) = S_{q^{'}}(\mathcal{X})$, $S_q(V) = S_q(\mathcal{X})$, and $|V| \leq 3$. Then, $R_{q^{'}}^2 \geq R_q^2$.*

Theorem 6.6 *For $C = 1$, the quantity*

$$\frac{\max_{i,j} \|\phi(\vec{x}_i) - \phi(\vec{x}_j)\|_q}{2}$$

is a monotonically nondecreasing lower bound on R as a function of q.

Corollary 6.3 *If $\vec{x}$ and $\vec{y}$ are the maximally separated pair of points in the data space, i.e. $(\vec{x}, \vec{y}) = \arg\max_{i,j} \|\vec{x}_i - \vec{x}_j\|$, then, for $C = 1$, the value*

$$\frac{1 - K_q(\vec{x}, \vec{y})}{2}$$

is a monotonically nondecreasing lower bound on R^2 as a function of q.

[23] Lee and Daniels (2005b) assume a Gaussian kernel, but this more general assumption is enough, because we only need that $K(\vec{x}, \vec{x}) = 1$ (see subsection 6.7.2).

Theorem 6.7 *For $1/n < C \leq 1$, the quantity*

$$\frac{\min_{i,j} \|\phi(\vec{x}_i) - \phi(\vec{x}_j)\|_q}{2}$$

is a monotonically nondecreasing lower bound on R as a function of q.

Corollary 6.4 *If $\vec{x}$ and $\vec{y}$ are the minimally separated pair of points in the data space, i.e. $(\vec{x}, \vec{y}) = \arg\min_{i,j} \|\vec{x}_i - \vec{x}_j\|$, then the value*

$$\frac{1 - K_q(\vec{x}, \vec{y})}{2}$$

is a monotonically nondecreasing lower bound on R^2 as a function of q.

Note that these lower bounds are all equal to zero for $q = 0$. They are all equal to $1/2$ for $q = +\infty$.

6.4.1.2 The secant algorithm

To generate the kernel width list, a numerical secant algorithm is used. In a secant method, the goal is to find an abscissa value v for which the curve to approximate intersects a horizontal axis. This is performed by generating a sequence of abscissa values that converges to v. Each subsequent value is given by the intersection of the aforementioned horizontal axis with the line through the curve points corresponding to the two preceding values in the sequence.
In our case we have q values as abscissa values, R^2 values as ordinate values and the line $R^2 = 1 - 1/n$ plays the role of the horizontal axis. Since $R^2 = 1 - 1/n$ is an asymptote (see Corollary 6.2), an R^2 curve approximated with this algorithm never intersects such a horizontal axis for finite values of q. Therefore, we are able to generate a potentially infinite increasing sequence of q values. The first two kernel width values are the origin $q_0 = 0$ and the initial kernel width value of Equation 6.26 which we denote by $q_1 = 1/\max_{i,j} \|\vec{x}_i - \vec{x}_j\|^2$. The choice of q_0 as starting q value is justified by Lemma 6.7. Let $R_0 = 0$ be the radius value corresponding to q_0 and let R_1 be the radius value corresponding to q_1. Let $P_0 \equiv (q_0, R_0)$ and $P_1 \equiv (q_1, R_1)$ be two points. The next q value, say q_{new}, is obtained by finding the intersection point between the line through the points P_0 and P_1 and the line $R^2 = 1 - 1/n$. For each additional q value, the associated R^2 is computed by finding the related MEB and then calculating the its radius.
Let q_{older} and q_{old} be the last two q values calculated and let $P_{older} \equiv (q_{older}, R_{older})$ and $P_{old} \equiv (q_{old}, R_{old})$ be two points, where R_{older} and R_{old} are the radii corresponding to q_{older} and q_{old} respectively. Each q_{new} value is obtained by finding the intersection point between the line through the points P_{older} and P_{old} and the line $R^2 = 1 - 1/n$.
Even though we have previously said that the algorithm can generate an infinite sequence of abscissa values, i.e. q values, in practical implementation things are slightly different. The algorithm stops when the slope of the last secant line is close to zero, i.e. such a line is almost parallel to the $R^2 = 1 - 1/n$ (which in turn is parallel to the abscissa axis). We have to choose a threshold ϵ to decide what

Algorithm 14 Secant-like kernel width generator

1: **procedure** SECANTWG($\mathcal{X}, C$)
2: $q_{older} \leftarrow 0$; $q_{old} \leftarrow \frac{1}{\max_{i,j} \|\vec{x}_i - \vec{x}_j\|^2}$
3: ▷ KM is the kernel matrix, β are the Lagrangian multipliers
4: $(KM, \beta) \leftarrow$ solveSVDD($\mathcal{X}$, q_{old}, C)
5: $R_{older} \leftarrow 0$; $R_{old} \leftarrow$ getSphereRadius(KM, β)
6: QList $\leftarrow \emptyset$
7: stop $\leftarrow$ **false**
8: **repeat**
9: $P_{older} \leftarrow (q_{older}, R_{older})$; $P_{old} \leftarrow (q_{old}, R_{old})$
10: secantLine $\leftarrow$ line through P_{older} and P_{old}
11: $P_{new} \leftarrow$ intersectLines(secantLine, $R^2 = 1 - 1/n$)
12: $q_{new} \leftarrow$ abscissa(P_{new})
13: $(KM, \beta) \leftarrow$ solveSVDD($\mathcal{X}$, q_{new}, C)
14: $R_{new} \leftarrow$ getSphereRadius(KM, β)
15: **if** $R_{new} < R_{old}$ **then** ▷ monotonicity broken
16: $q_{new} \leftarrow$ getMonotonicQ(q_{old}, q_{new}, KM)
17: $(KM, \beta) \leftarrow$ solveSVDD($\mathcal{X}$, q_{new}, C)
18: $R_{new} \leftarrow$ getSphereRadius(KM, β)
19: **end if**
20: QList $\leftarrow$ QList $\cup \{q_{new}\}$
21: **if** $(R_{new} - R_{old})/(q_{new} - q_{old}) < \epsilon$ **or** $\forall i = 1, 2, \cdots, n, \beta_i > 0$ **then**
22: stop $\leftarrow$ **true**
23: **else**
24: $R_{older} \leftarrow R_{old}$; $R_{old} \leftarrow R_{new}$; $q_{older} \leftarrow q_{old}$; $q_{old} \leftarrow q_{new}$
25: **end if**
26: **until** stop
27: **return** QList
28: **end procedure**

"close to zero" means, e.g. stop when $(R_{new} - R_{old})/(q_{new} - q_{old}) < 10^{-6}$, where the ϵ is 10^{-6}. A second stopping criterion is used too: we stop the algorithm when all data points become support vectors, because in such a case the number of clusters is equal to the size n of the dataset $\mathcal{X}$ and further q values will not produce any meaningful clustering.
If we take a look at Algorithm 14, we note the lines $15 : 19$. These lines face the monotonicity violation. Although it is rare, the monotonicity of R^2 may be broken by some kernel width values. To ensure the monotonicity, we check the R_{new} value: if it is less than the previous one, then we have to fix the q_{new} value. There are several techniques to do this. The first one is called *support vector permanence* and it requires that each support vector for q_{old} is also a support vector for q_{new}. This is motivated by Theorem 6.5. Support vector permanence also requires that no new BSVs be added for q_{new}. The support vector permanence could be achieved by adding some constraints to the QP problem of the SVDD for q_{new}

- $\forall \vec{x}_i \in SV_{q_{old}}, \beta_i > 0$

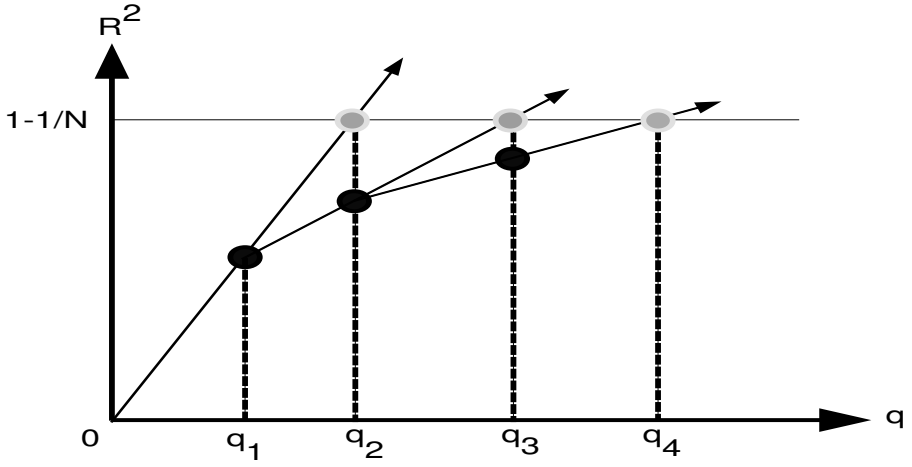


Figure 6.12: *An example of how the secant-like kernel width generator works. The curve through all fulfilled black dots is the approximation of the R^2 curve. The sequence of q values is $q_1, q_2, q_3, q_4, \cdots$. In the figure, N is the size of the input dataset (Lee and Daniels, 2005a, fig.1).*

- $\forall \vec{x}_i \notin BSV_{q_{old}}, \beta_i < C$

where $SV_{q_{old}}$ and $BSV_{q_{old}}$ are the set of the SVs and BSVs for q_{old}, respectively. Support vector permanence never failed in practice, but the monotonicity in this case is only a conjecture, so it could fail. Therefore, another technique is available in such a case. We distinguish two sub-cases. If $R_{old} < 1/2$, the monotonicity is enforced by using the lower bound of the Corollary 6.4, which allows to generate a piecewise linear approximation to the lower bound curve and sets q_{new} to the q value at which a horizontal line through R_{old} intersects the approximation. If $R_{old} \geq 1/2$, we need a brute force approach, whatever it is.
The secant method samples the R^2 curve at a frequency that is a function of the slope of the curve approximation. Therefore, it generates the most q values where the R^2 curve rises sharply. An additive Δq change induces an exponential decrease in each kernel width value, which, in turn, causes an exponential decrease in the cosine of each angle between two images on the unit ball in feature space. Empirical observations suggest that the R^2 curve exhibits overall logarithmic behavior and changes in clustering results occur most often when changes in R^2 values are most relevant.

6.4.1.3 The length of the kernel width list

The length of the kernel width list is useful in order to know, at each step, the overall complexity of the *secant-like kernel width generator* algorithm. An estimate of the kernel width list length, say Q, was proposed. It results[24] in

[24]The proof is in Lee and Daniels (2005a,b)

$$Q \approx \lg \max_{i,j} \|\vec{x}_i - \vec{x}_j\|^2 - \lg \min_{i,j} \|\vec{x}_i - \vec{x}_j\|^2 \tag{6.46}$$

The equation above depends only on spatial characteristics of the dataset and not on the dimensionality of the dataset itself or the number of points.

6.4.1.4 Softening strategy

The softening strategy is the first of our own contributions to the *secant-like kernel width generator*.
We have observed that some important kernel width values were skipped by the algorithm. Let $QList = \{q_i\}_{i=1,2,\cdots,Q}$ be the set of the kernel width values generated for a given problem. In several experiments we have further tuned such values to check whether the q values in $[q_i, q_{i+1}]$ (for some $i = 1, 2, \cdots, Q$) were relevant to the problem. Our empirical observations on a number of experiments have revealed that a general interval $[q_i, q_{i+1}]$ is often too large; consequently, the algorithm does not output some important values that would lead to very good clustering results.
Therefore, we have modified the original algorithm with a simple heuristics, which we called *softening strategy*. It consists of introducing a *softening parameter* s, $0 < s \leq 1$, that shades the q_{new} value by means of the sq_{new} product, added between the lines 19 and 20 of the Algorithm 14. If $s = 1$ the algorithm performs like the original version. In several experiments, we set $s = 0.5$ and obtained very good results.
Note that such a heuristics can potentially increase the probability of a monotonicity violation to occur. Anyway, in our tests this has never happened.

6.4.1.5 Wrapping the Kernel Width Generator around the SVC loop

Other problems derive from the use of this strategy to create a list of kernel width values before running the SVC. That is, the procedure is conceived to produce a list of suitable kernel width values; such values are then used to execute the SVC at different scales of details. This approach presents two drawbacks. First, we solve twice the same instance of an SVDD problem related to a q_i value: the first time when we have to compute q_{i+1} during the kernel width generation process, and the second time when we use the q_i value in the SVC.
The second drawback is that the number of generated q values is often a surplus, i.e. the SVC usually finds the desired clustering in a number of steps that is definitely less than the number of the kernel width values produced by the secant-like algorithm. This results in a waste of time to solve a number of useless SVDD instances.
Therefore, our idea is to wrap the kernel width generator around the SVC main loop. In such a way, we address both of the aforesaid issues: each SVDD instance is solved once and only the really useful SVDD instances are solved. The latter is achieved thanks to the stopping criteria used to halt the SVC loop, that usually breaks the execution when the most suitable clustering results are found.

This is our second contribution to the secant-like strategy for kernel width generation.

6.4.1.6 Complexity

Let us first analyze the computational complexity of the original version of the algorithm.
The computational complexity of the intersection of two lines is $O(1)$, because in practical implementations we take a pair of points from each line and calculate the intersection point. Next, we have the procedure for finding the MEB, which is the same procedure to perform the cluster description in the SVC. Therefore, we already know its theoretical complexity is $O(n^3)$ (see subsubsection 6.1.5.1). The calculation of the sphere radius costs $O(n_{sv}^2)$.
We conclude that the overall worst-case running time is $O(Q(n^3 + T))$, where Q is the estimate of the kernel width list length (see subsubsection 6.4.1.3) and T is the running time[25] of the procedure to enforce the monotonicity. Generally, the *support vector permanence* never fails, therefore T is usually equal to $O(n^3)$.[26]
The brute force strategy is very uncommon, therefore we can conclude that the running time of the monotonicity enforcement procedure can be absorbed in the running time of the procedure for solving the SVDD problem.
Wrapping the secant algorithm around the SVC process substantially changes the facts. In the first place, we do not need the Q estimate anymore, because the number of generated kernel width values is now equal to the number of SVC iterations. In fact, a new kernel width value is generated only if the SVC does not fulfill its stopping conditions; such conditions are usually achieved in a number of iterations that is definitely less than and independent of Q. Moreover, since the solution of the SVDD problems is now provided by the SVC process,[27] the secant-like algorithm adds only a cost of $O(n_{sv}^2 + T)$ per iteration. Because of the aforementioned considerations about the monotonicity enforcement procedure, such a cost can be absorbed into the running time complexity of an SVC step, i.e. the worst-case running time complexity of the SVC is unchanged (see subsubsection 6.1.5.3).
Finally, the introduction of our *softening strategy* does not affect the worst-case running time complexity. Even better, in some cases it may also reduce the actual number of iterations.

6.4.1.7 Conclusion

The *secant-like kernel width generator* provides a robust means to explore the kernel width parameter. It is the first work in this direction and is a great contribution as it makes the SVC algorithm feasible in practice.

[25] The value of T can be different according to the monotonicity enforcement strategy employed for each monotonicity violation, and according to the number of monotonicity violations that occur. Usually, the monotonicity violations are rare.

[26] We recall that the support vector permanence solves another SVDD problem with a couple of additional constraints.

[27] The secant-like generator use the results from the SVDD solved by SVC.

Since we discovered some drawbacks in the process, we proposed some contributions to improve it. The first one is the *softening strategy* heuristics, which enable us to generate some useful kernel width values that the original procedure leaves unexplored. The second improvement is the *wrap* of the generator algorithm around the SVC main loop, which drastically reduces the computational effort and the number of the uselessly solved SVDD instances. Finally, we have revealed that the secant-like kernel width generator can work with all normalized kernels which are suitable for the SVC (see section 6.7), and not only with the Gaussian kernel as stated by Lee and Daniels (2005b).[28]
We have successfully used this secant-like strategy in our experiments.

6.4.2 Angle-decrement kernel width generator

The *angle-decrement kernel width generator* is a variant of the previously described secant-like generator (Hartono and Widyanto, 2007).[29] The main idea of this method is based on the decreasing angle in secant-like strategy.
First, the initial kernel width value is calculated. With the secant-like method, for this value of q we calculate the line that intersects the line $R^2 = 1 - 1/n$. In this case we consider the angle between such a secant line and the abscissa axis; it is obtained with the following calculations

$$q_1 = \frac{1 - \frac{1}{n}}{\tan\theta_1} \tag{6.47}$$

$$\tan\theta_1 = \frac{1 - \frac{1}{n}}{q_1} \tag{6.48}$$

$$\theta_1 = \arctan(\tan\theta_1) \tag{6.49}$$

The subsequent q value is calculated by decrementing the angle θ and calculating $\tan\theta$. Generally, we have

$$q_Q = \frac{1 - \frac{1}{n}}{\tan\theta_Q} \tag{6.50}$$

and

$$\tan\theta_Q = \tan(\theta_{Q-1} - \Delta_\theta) \tag{6.51}$$

where Δ_θ is the quantity deducted to an angle for obtaining the next one.
First, the q_1 is calculated, then the value of $\tan\theta_1$ is computed and θ_1 is obtained by $\arctan(\tan\theta_1)$. To get the next θ_2 value, the value of the current θ_1 is decremented by a Δ_θ value. Finally, we can calculate the next q_2 value by dividing $1 - 1/n$ with $\tan\theta_2$.

[28] As we will see in the sequel, this is also supported by the experimental results.

[29] Like the secant-like method, this method assumes to generate kernel width values for the Gaussian kernel. Anyway, for the same reasons previously explained, the validity of the method still holds for any normalized kernel which is appropriate for the SVC.

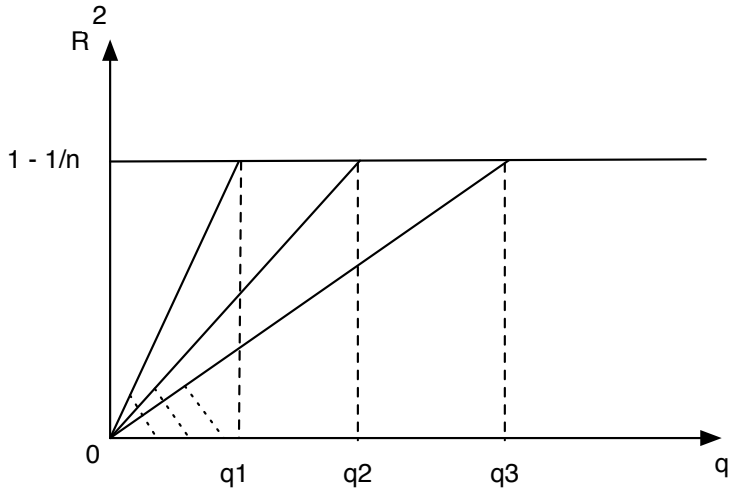


Figure 6.13: *An example of how the angle decrement kernel width generator works. The angle is decreased at each step. The sequence of q values is $q_1, q_2, q_3, \cdots$. In the figure, n is the size of the input dataset (readapted from Hartono and Widyanto (2007, fig.4)).*

The angle-decrement method computes q_i from q_{i-1} without calculating the related radius value, i.e. no SVDD computation is needed as in the secant-like method.
The process of generation guarantees that

$$\forall i = 1, 2, \cdots, Q-1, \theta_i > \theta_{i+1} \tag{6.52}$$

$$\forall i = 1, 2, \cdots, Q-1, \tan\theta_i > \tan\theta_{i+1} \tag{6.53}$$

and

$$\forall i = 1, 2, \cdots, Q-1, q_i < q_{i+1} \tag{6.54}$$

Since this method does not compute R^2 value for each q value anymore, the process results less computational demanding than the secant-like algorithm.

6.4.2.1 Conclusion

Currently, this method is just a proposal and the main drawback is the Δ_θ value. In fact, it is not clear how to appropriately determine this value, in order to obtain only relevant kernel width values.
The advantage is the low computational effort of this strategy. A robust way to decrement the angles could make this method a great alternative to the secant-like method.
However, in this thesis we showed how to eliminate the problem of the additional and useless SVDD computations from the *secant-like kernel width generator* without loosing the robustness of this algorithm (see subsubsection 6.4.1.5). This could make the development of the *angle-decrement kernel width generator* useless.

6.4.3 Other works on kernel width exploration

Since the kernel width values close to the initial kernel width value could lead to a single, all-inclusive, cluster, an alternative way to determine the initial kernel width value was proposed. Its name is *Empty circle method* (Lee and Daniels, 2005b, sec. 4.2.5). It selects a distance that is the same as the radius of the largest empty circle in the convex hull of the data. The *largest empty circle* is defined as the largest circle containing no points and whose center lies inside the convex hull of the data. This circle has at least two points on its boundaries.

However, the computational complexity of this method increases as the dimensionality of the dataset grows, because it has $O(n^{\lfloor d/2 \rfloor})$ worst case running time complexity, where d is the dimensionality of the data.

6.5 Soft margin constant estimation

The work on the estimation of the soft margin constant C is practically inexistent. A first embryonal idea was proposed in Lee and Daniels (2005a, sec. 3): the β values contain useful information to distinguish outliers from non-outliers. For example, if we compute the average β value of all data points across all SVC iterations and count the number of data points having a larger β value than the average β value, then we can use that number as an approximate number of outliers in the data set. For instance, if we have a dataset with $n = 500$ and obtain an estimate of 107 outliers, we can set $C = 1/107 = 0.009$.

However, in our experiments such an estimation has worked very poorly. Furthermore, the computational effort to obtain such an estimation of C is definitely unacceptable.

So, we have elaborated a first alternative which is a modified version of the above strategy: we run once the SVC process with the initial kernel width value and $C = 1$. Then, we estimate C as follows: we calculate an average β value across all points in the dataset and consider outliers only those points having a β value larger than the average. This modification drastically reduce the computational complexity of the original above idea. Anyway, also in this case we have got poor results.

Finally, we have made an empirical observation about the C value. We have observed that $C = 10/n$ is a good choice for most problems. From Equation 6.21 we know that $p = 1/nC$ is an upper bound for the fraction of outliers, so setting $C = 10/n$ is equal to set such an upper bound to $1/10$, i.e. the number of outliers can be at most the 10% of the dataset cardinality.

This estimate yielded very good results in most of our experiments. In the remaining cases, a little tuning was needed (by increasing or decreasing the estimated C value).

In our experimental phase, the last result allowed us to be free from the C parameter tuning, which in turn resulted in a time saving.

6.6 Stopping criteria

The original stopping criterion proposed for the SVC was to establish a threshold on the maximal number of support vectors (Ben-Hur et al., 2001, sec. 4.2). In fact, the number of support vectors determines the number of clusters and the meaning of the clustering results: when the number of support vectors becomes too large, the clusters become too many and too small, leading to a meaningless partitioning. However, same problems occur if the number of support vectors it too small: we obtain a small number of very large clusters.

Another possible stopping criterion was proposed in Ben-Hur et al. (2001): the stability of cluster assignments over some range of the two hyper-parameters.

These two proposed stopping criteria have some serious drawbacks. The first one relies on a factor (the number of SVs) which highly depends on the spatial and statistical characteristics of the dataset at hand, i.e. the threshold to set as SVs upper bound is different for each problem we have to face. Moreover, if we tune the soft margin parameter C, the number of clusters and the smoothness of the cluster boundaries change too. Hence, if we state that the number of total support vectors must be at most the 5% of the total points, this statement assumes different meanings according to the ratio between the free support vectors (i.e. the support vectors on the cluster boundaries) and the bounded support vectors (i.e. the support vectors which are turned in outliers). After all, it is absolutely not clear how to determine such a threshold.

The stability of cluster assignments is meaningless too. The cluster assignments can be stable over some range of kernel width values, but a better partitioning can be found also after these stable assignments.

Therefore, let us analyze some alternatives.

6.6.1 Stopping criteria based on number of clusters

We have stated at the beginning of this chapter that the SVC inherits the advantages of a hierarchical clustering algorithms, such as the ability of making no assumptions on the number of clusters. In fact, one of the main advantages of the SVC is the ability of discovering the most meaningful partitioning. Anyway, building a stopping criterion on the requested number of clusters could be useful when we make experiments with labeled data, in order to have a direct comparison for evaluating the results. For instance, if we have just improved the algorithm with new features and want to test it on labeled data, such a stopping criterion could be useful, because we know the number of clusters we want to find.

We have built such a stopping criterion. Once the algorithm achieves the requested number k of clusters for the first time, it continues the process until the number of clusters changes or the number of singleton clusters increases. The singleton clusters (or, equivalently, clusters with a very low cardinality) are usually neglected during the clustering process. Besides, in the SVC such clusters usually contain outliers: one more reason to reject them.

We have tested this criterion in experiments with labeled data and it worked very fine every time.
Anyway, such a criterion is not generally useful for a clustering algorithm like SVC, because one of its advantages is the ability to discover the number of clusters by itself.

6.6.2 Stopping criteria based on kernel width

Lee and Daniels (2005a) provided us with a robust way to generate a list of suitable kernel width values, allowing us to analyze the clusters at different levels of detail. Thanks to our improvement in subsubsection 6.4.1.5 we are able to exploit new stopping criteria. In the first place, we can use one of the stopping criteria employed by the *secant-like kernel width generator*: if the slope of the generated secant line is close to zero, we stop the generation process, because the last secant line was very close to the asymptote $R^2 = 1 - 1/n$ (see subsubsection 6.4.1.2). We can also stop when we reach the maximal useful kernel width (see Equation 6.27).

6.6.3 Stopping criteria based on relative evaluation criteria

One of the most promising ideas for developing a robust stopping criterion relies on the *relative criteria* for the evaluation of the clustering results. The relative criteria (see subsection 3.3.3) usually consist of indices that evaluate the quality of the partitioning. Such indices are usually used to evaluate different clustering results obtained by the same clustering algorithm which we have executed with different parameter settings. The best index value indicates the best partitioning. For example, the C-index assigns to the best clustering results the lowest index value.
To employ a relative criterion for developing an SVC stopping criterion, we propose a simple strategy. The execution scheme of the SVC is a divisive scheme, which provides at each iteration a number of clusters greater than or equal to the number of clusters provided in the previous iteration. Most likely, we will see an increasing quality of clustering for a number of iterations, then the quality will begin to decrease till becoming totally meaningless. By means of a relative criterion we can stop when the quality of the clustering starts to decrease.
Although we can exploit one of the available relative criteria for the evaluation of clustering results, an *ad hoc* index for the SVC algorithm would be a better choice. A validity index which relies on intrinsic peculiarities of the SVC has been developed (Chiang and Wang, 2004, 2006; Wang and Chiang, 2008).

6.6.3.1 SVC validity measure

The SVC validity measure is defined as the ratio between the overall compactness of the clusters and the minimum divergence between clusters (Chiang and Wang, 2004, 2006; Wang and Chiang, 2008). Let us provide some definitions.

Definition 6.15 (Intracluster distance) *Let $\mathcal{C}_i$ be the i-th cluster. The intracluster distance D_i is defined as the overall distance between the data points inside the cluster $\mathcal{C}_i$*

$$D_i = \frac{1}{n_{\mathcal{C}_i}(n_{\mathcal{C}_i} - 1)} \sum_{\vec{x}_j, \vec{x}_k \in \mathcal{C}_i} \|\vec{x}_k - \vec{x}_j\|^2 \tag{6.55}$$

where $n_{\mathcal{C}_i}$ is the number of elements in the cluster $\mathcal{C}_i$.

Definition 6.16 (Overall clustering compactness) *The overall clustering compactness measure D is defined as*

$$D = \max_i D_i \tag{6.56}$$

Definition 6.17 (Intercluster divergence) *Let $\mathcal{C}_i$ and $\mathcal{C}_j$ be two clusters. The intercluster divergence S_{ij} is defined as the distance between the support vectors into the cluster $\mathcal{C}_i$ and those ones into the cluster $\mathcal{C}_j$.*

$$S_{ij} = \min_{\vec{x} \in SV_i, \vec{y} \in SV_j} \|\vec{x} - \vec{y}\|^2 \tag{6.57}$$

where $SV_i \subseteq \mathcal{C}_i$ and $SV_j \subseteq \mathcal{C}_j$ are the sets of support vectors for the clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ respectively.

Definition 6.18 (Overall clusters divergence) *The overall clusters divergence measure S is defined as*

$$S = \min_{i,j} S_{ij} \tag{6.58}$$

From above definitions we can define the SVC validity measure as follows.

Definition 6.19 (Validity measure) *The validity measure G is defined as the ratio of the compactness measure D to the divergence measure S*

$$G = \frac{D}{S} \tag{6.59}$$

The formulation of the SVC validity measure clearly relies on intrinsic SVC properties. We recall that the SVC can be considered as a *density based clustering algorithm*. In fact, the validity index above is justified by a practical observation: we expect that clusters should be as dense as possible and the distance between clusters should be as far as possible. Therefore, the compactness measure D is expected to be smaller and the divergence measure S is expected to be larger. A small G indicates a large compactness and a large intercluster distance. This implies that the smaller the G the higher the quality of the clustering results.

6.6.4 Conclusion

In this section we have provided some alternatives to the original stopping criteria proposed by Ben-Hur et al. (2001). The criteria in subsection 6.6.1 and in subsection 6.6.2 are our ideas. We have also proposed a simple strategy to employ a validity index for developing a stopping criterion (see subsection 6.6.3); such a strategy has never been employed before. In fact, the authors of the validity index in subsubsection 6.6.3.1 did not use such an index to stop the clustering process, but only to *a posteriori* evaluate the clustering results.
It is important to note that we are not constrained to use a single stopping criterion. A combination of several (compatible) criteria can be employed, which is usually a common choice in practical implementations.

6.7 Mercer kernels for Support Vector Clustering

Although Support Vector Machines (SVMs) works well with several kernel functions (Smola et al., 1998) in case of classification and regression, we have seen that the Gaussian kernel (see Equation 6.18) is almost exclusively employed with the SVC. Above all, this is due to the SVDD: it seems to have a remarkable behavior only with the Gaussian kernel. Next, the Gaussian kernel is the most reliable and flexible kernel, which yield good performance in several applications and it should be considered especially if no additional knowledge about the data is available (Smola et al., 1998). Furthermore, in case of high BSV regime, it can be shown that the Gaussian kernel is the only kernel for which the number of clusters is a monotonically nondecreasing function of kernel width q. Anyway, starting from experiments, we have studied the application of different kernels to the problem of the Support Vector Clustering (SVC).

6.7.1 Basic kernel requirements

Since the SVC needs to analyze data at different levels of detail, the first requirement that a kernel have to satisfy is that capability, i.e. it has to have a scale-control parameter like the kernel width.
For example, in literature Equation 6.18 is often referred as Gaussian *kernels*, because we have a different behavior depending on the value of the kernel width (Cristianini and Shawe-Taylor, 2000; Smola et al., 1998). From this viewpoint, all kernels which have a scale-control parameter like the kernel width can be referred as a family of kernels instead of a just single kernel. Therefore, what we need is kernel families to employ with the SVC.
On the other hand, such a control parameter can be introduced potentially in any kernel. While in the Gaussian kernel the width has a clear mathematical meaning (it is a function of the variance), in other kernels the width simply acts as a smoothing parameter. In reality, it plays exactly the same role of the variance in the Gaussian kernel. In fact, in practical implementations (e.g. the LIBSVM (Chang and Lin, 2007)), several kernels have an additional parameter generally

called "width" or *gamma*, notwithstanding their original formulations did not include it. In such a way, we turn a single kernel in a family of kernels.
However, this does not imply that kernels which produce poor results with SVC can change its performance thanks to this new parameter. In fact, the polynomial kernel in LIBSVM have the kernel width parameter, but still works poorly as stated by Tax (2001).
The second important requirement is the normalization. A normalized kernel is needed for several reasons: the first one is that the secant-like width generator assumes a normalized kernel. The CCL also does the same. A secondary reason is that a normalized kernel makes the SVDD equivalent[30] to the *One Class SVM*, which is more likely to found in practical implementations than the SVDD (Chang and Lin, 2007).

6.7.2 Normalized kernels properties

A kernel $K(\cdot)$ is said to be normalized when it automatically scales the data to unit norm, that is

$$\forall x \in \mathcal{X}, K(\vec{x}, \vec{x}) = 1 \tag{6.60}$$

Some kernels are "self-normalized", like the Gaussian, the Exponential, the Laplacian ones (see subsubsection 2.6.1.1). Anyway, we can always build a normalized version of any kernel, by means of following law

$$K_1(\vec{x}, \vec{y}) = \frac{K(\vec{x}, \vec{y})}{\sqrt{K(\vec{x}, \vec{x})K(\vec{y}, \vec{y})}} \tag{6.61}$$

where $K(\cdot)$ is a kernel and $K_1(\cdot)$ is its normalized version.
Equation 6.60 implies that the data are implicitly scaled to have a unit norm. Therefore, the entire data space is embedded onto the surface of the unit ball in feature space $\mathcal{F}$, which in turn implies that the MEB is never larger than the unit ball.
Equation 6.60 also implies that $K(\vec{x}, \vec{y}) = \cos\theta$, where θ is the angle between $\phi(\vec{x})$ and $\phi(\vec{y})$. Furthermore, we have

$$\forall \vec{x}, \vec{y} \in \mathcal{X}, 0 \leq K(\vec{x}, \vec{y}) \leq 1 \tag{6.62}$$

Therefore, no pair of points in feature space can have angles greater than $\pi/2$. This means that the image of the data space can be rotated so that it lies within an *octant* of the feature space.
Finally, when the kernel $K(\cdot)$ is a normalized kernel, Equation 6.15 can be rewritten as follows

$$d_R^2(\vec{x}) = 1 - 2\sum_{k=1}^{n_{sv}} \beta_k K(\vec{x}_k, \vec{x}) + \sum_{k=1}^{n_{sv}} \sum_{l=1}^{n_{sv}} \beta_k \beta_l K(\vec{x}_k, \vec{x}_l) \tag{6.63}$$

[30] See Appendix A.

6.7.3 Support Vector Clustering and other kernels

In some of our experiments (see chapter 9) we have tried some different kernels to understand what type of kernel functions could be suitable for the SVC and to understand also what is the behavior of the cluster description stage with a different mapping in feature space.
However, this is not the first attempt to try kernels other than the Gaussian one. A first test with the Laplacian kernel can be found in Ben-Hur et al. (2000a), whereas some researchers have developed an *ad hoc* kernel for their own application domain (Ling et al., 2006; Ling and Zhou, 2006; Ping et al., 2005).

6.7.3.1 Laplacian kernel

The Laplacian kernel

$$e^{-q|\vec{x}-\vec{y}|} \tag{6.64}$$

has the same shape of the the Laplacian distribution, even though it should not be confused at all with the latter. It satisfies the normalization requirement.
The Laplacian kernel was the first kernel we have tried as it was also employed in one of the early works about the SVC (Ben-Hur et al., 2000a). Even though the use of this kernel may not always be desirable because it is far not as smooth as the Gaussian one in the data description, we have observerd some remarkable behaviors of such a kernel function under certain conditions.
Above all, the SVC with the Laplacian kernel has shown fine performance dealing with some datasets where the data has been scaled to have a unit norm. However, this is not a systematic behavior: in other cases the kernel worked poorly also with rescaled data or worked fine with non-scaled ones.[31]
The Laplacian kernel has shown a remarkable behavior (but under some particular conditions). In some cases it has performed better than the Gaussian kernel. Finally, we have seen that the number of clusters is not a nondecreasing function of the kernel width anymore. In fact, after a number of iterations, the SVC yields a single all-inclusive cluster again. Anyway, thanks to the stopping criteria, SVC is able to halt before such a behavior is shown.

6.7.3.2 Exponential kernel

The Exponential kernel is apparently similar to a Gaussian kernel, but it lacks of an important part, i.e. the distance is not squared

$$e^{-q\|\vec{x}-\vec{y}\|} \tag{6.65}$$

[31] A couple of experiments also revealed that the data described with a Laplacian kernel can probably labeled better if we use an $L1$ distance ($L1$ distance is the same distance used by the Laplacian kernel) to check the similarity between vectors in the labeling stage. Anyway, due to some practical limitations, we were not able to replace all the $L2$ distance instances with $L1$ ones in order to have a homogenous situation. Therefore we stopped the experiments in this direction, but such a behavior should be further investigated.

The Exponential kernel satisfies the normalization requirement. It usually works fine with the same data the Gaussian kernel does, and in some cases it worked even better. However, also in this case, if we remove the SVCÂ stopping criteria, we can see that the number of clusters is not a nondecreasing function of the kernel width anymore.

6.7.3.3 Radial Basis Function kernels

The Gaussian kernel, the Laplacian kernel and the Exponential kernel are all Radial Basis Function (RBF) kernels (Lin, 2005). For this reason, in future, it worths to test also other RBF kernels, such as the *Stump* kernel and the *Perceptron* kernel (Lin, 2005).

6.7.4 Conclusion

Although we have performed a limited number of experiments with different kernel functions, we have verified some important things. First, we have seen that with kernels other than the Gaussian one the number of clusters is not a nondecreasing function of the kernel width q anymore. But we have also verified that this is not as problematic as we thought. In fact, the monotonicity is kept for several iterations before it is broken, and the SVC generally halts before the monotonicity is violated. Anyway, the SVC always shows the same behavior when such a monotonicity is broken, i.e. it can not separate clusters anymore. If we are using a stopping criterion based on a validity index, this means that we will stop exactly when the monotonicity will be broken, because the validity index will be poor for a trivial clustering.
In the second place, we have verified that we are not constrained to the Gaussian kernel, but we can employ different kernels, sometimes only under certain conditions. The problem in these cases is that we must know the application domain very well so that we can choose a really suitable kernel.[32] In fact, the main difference between the Gaussian kernel and all other kernels is that the former still remains the kernel with the best performance on average, because it works with several kind of data both scaled or not scaled. This is an intrinsic peculiarity of this kernel which has the same success also in classification, regression and other learning problems (Smola et al., 1998).
Finally, we unveiled some basic requirements: the kernels should be normalized and must have a kernel width parameter.

6.8 Scaling down the overall time complexity

To reduce the overall time complexity of the SVC we can tackle the problem from three different sides. First, we can improve the speed of the cluster description

[32] Early works on the automated selection of the most suitable kernel are available for the SVM classification (Ali and Smith-Miles, 2006). Most likely, such works could be extended also to other types of SVM applications.

stage by optimizing the SVM training process. In the second place, we can introduce some preprocessing steps. Finally, we can reduce the time complexity of the cluster labeling stage.[33]

6.8.1 Improving the SVM training

The QP problem (which the SVM training is based on) is a very interesting optimization problem. Several optimization algorithms has been proposed to efficiently solve this QP problem both from a time-consuming perspective and a space-consuming perspective: chunking (Vapnik, 1982), Successive Over-relaxation (SOR) (Burges, 1998), Sequential Minimal Optmization (SMO) (Platt, 1998), SVM^{light} (Joachims, 1998).
In practice, state-of-the-art implementations typically have a training time complexity that scales between $O(n)$ and $O(n^{2.3})$ (Platt, 1999; Tsang et al., 2005). This can be further driven down to $O(n)$ with the use of a parallel mixture (Collobert et al., 2002). The space complexity can be also reduced to $O(1)$ at the cost of a decrease in efficiency (Ben-Hur et al., 2001, sec. 5).

6.8.1.1 Spatially chunking for Support Vector Clustering

In Ban and Abe (2004) we found a modified version of the previously mentioned *chunking*, which is designed for the SVC. This work starts from the observation that despite the immense effort researchers have taken to speed up the SVM training, a problem such as pixel clustering for an image with moderate size will keep a machine running for hours.
Therefore, the authors have proposed an improvement of the original chunking method, called *Spatially Chunking*. The spatially chunking strategy takes the spatial information into account. Obviously, this work was made with a specific application domain in mind, the spatial data analysis (see subsection 1.1.4), therefore it is not applicable in any case.

6.8.1.2 Support Vector Machines reformulation

The previously mentioned complexities are only empirical observations and not theoretical guarantees. For reliable scaling behavior to very large datasets, a specific algorithm should be developed: it should be asymptotically efficient in both time and space. A key observation is that the practical SVM implementations only approximate the optimal solution of the QP problem by means of an iterative strategy. For example, in SMO and SVM^{light} the training process stops when the Karush-Kuhn-Tucker (KKT) conditions are fulfilled within a tolerance parameter ϵ. The experience indicates that suboptimal solutions are often good enough in practical applications. Therefore, an alternative way to speedup the training of SVMs can be the reformulation of the problem by employing the idea of the optimal solution approximation.

[33] Cluster labeling alternatives were yet analyzed in section 6.3, therefore we do not take them into account in this section.

In the last years some attempts have been made to reformulate the SVMs so that they were simpler to train, but without loosing the peculiarities that have made so famous and widespread such learning machines. Some works propose formulations which lead to the solution of a different problem (not a QP problem anymore). Some other reformulations were made to exploit different approximation algorithms for which the asymptotical complexity is not an empirical observation, but a theoretically provable fact. Two of the most remarkable works in this sense are the *Least Squares Support Vector Machines (LS-SVM)* (Suykens et al., 2002) and the *Core Vector Machines (CVMs)* (Tsang et al., 2007, 2005, 2006).
LS-SVM reformulation leads to solving a set of linear equations (the KKT system), which is simpler than solve a QP problem. A drawback of this formulation is the lack of sparseness in the solution, i.e. the Lagrangian multipliers for non-SV points are not zero as in the classical SVM solution. Anyway, a simple method for imposing sparseness is provided (Suykens et al., 2002).
CVMs reformulation exploits the close relationship between the SVM training and the Minimum Enclosing Ball (MEB) problem. It can be shown that the training of a Core Vector Machine (CVM) has $O(n)$ worst-case running time complexity and has $O(1)$ space complexity. By using probabilistic speedup methods, the actual training time can be further reduced.

6.8.2 Preprocessing data

An alternative way to reduce the time and space complexity of the SVC is to develop a preprocessing step for detecting a subset of points and use only those points for the cluster description stage. The idea is to remove from the training stage those points which do not contribute to the cluster description. Such a method was proposed by Nath and Shevade (2006) and is called *Heuristics for Redundant point Elimination*. By means of this heuristics, we can eliminate several points from the cluster description stage (so from the SVM training stage), drastically reducing the computational effort. The points not involved in the cluster description stage are successively reintroduced in the cluster labeling stage for a complete cluster assignment process.

6.9 Other works on Support Vector Clustering

Currently the SVC is still under heavy research to improve the overall clustering process by enhancing also the key characteristics of the SVC. There are several other works about the SVC and most of them are very early proposals. However, we wish to highlight two of the most recent works. The first one provides a robust way for calculating the entire regularization path of the SVDD and so of the whole clustering process (Hansen et al., 2007). This approach ensures that the pseudo-hierarchical clustering description of the SVC is complete.
A second noticeable work further improves the robustness of the SVC with respect to the noisy patterns and outliers (Yankov et al., 2007). It aims at outlining better interpretable clusters. The major modification is the addition of a local

reconstruction method to detect those points that are likely to deviate from the (unknown) distribution that has generated the data at hand. By means of this method (Mixture of Factors Analyzers (MFA)) a weighted system is built. The weight are applied to the penalty terms of the Lagrangian dual problem, i.e. to the slack variables. Such a weighting leads to determine the soft-margin tradeoff between the outliers and the accuracy of the SVDD learner, as well as to regularize the complexity of the induced decision boundary. Such a regularization results in smoother contours.

6.10 Contribution summary

In this section we present a summary of our own contribution. In the first place, a simple but effective modification to the Cone Cluster Labeling (CCL) was provided, which improves the labeling of the BSV points, especially in a high BSV regime.
In the second place we introduced the *softening strategy* in the *secant-like kernel width generator* algorithm, which allows to avoid skipping some crucial kernel width values. In our experiments, the softening strategy yielded effective results: lower number of SVC iterations and better accuracy in cluster description, and so in the final clustering results.
A second contribution to the secant-like kernel width generator is the wrapping of this process around the SVC loop. This drastically reduces the computational effort compared with a standalone kernel width generator algorithm which we are constrained to put as a preliminary step. Moreover, we can use the stopping criterion of the secant-like algorithm as an additional stopping criterion for the whole clustering process. Other contributions to stopping criteria has also been proposed, such as the stopping criterion based on the number of clusters (useful when we perform experimental sessions with labeled data) or the strategy that exploits a validity measure to halt the clustering process.
Another embryonal but effective contribution is the empirical observation about the bounded support vectors upper bound. Such an observation has eliminated the problem of the soft margin parameter tuning in our experiments and at the same time leads us to very good clustering results.

6.11 Conclusion

In this chapter a state-of-the-art clustering algorithm based on the SVM formalism was presented. This method has several advantages, which we have verified in our experiments described in chapter 9.
Above all, it produces a cluster description in feature space, so it can face well nonlinear separable problems. Moreover, the ability to analyze the data at different levels of detail make the SVC independent of the application domain: in fact, by tuning the hyper-parameters, we are able to accurately analyze the data structure, and so the clusters organization, whatever it is.

A distinctive characteristic is that it has no explicit bias of either the number or the shape of clusters. Therefore, it eliminates the problem of estimating the number of clusters (see section 3.4) as well as any constraint on the shape of clusters (see section 3.9).[34]
Another unique peculiarity of this clustering algorithm is the ability to deal with noise and outliers (see section 3.6 and section 3.8). By means of the soft margin parameter C, it is able to be robust with respect to noise. The points we call outliers, i.e. those points whose probability distribution is very different from the one of the majority of points, are excluded from the clusters without any interference in the clustering results. The same C parameter also allows to separate overlapping clusters. However, this behavior is not enough for datasets with very heavily overlapping clusters and the SVC can fail. In such cases, using a support vector description for each cluster may be an alternative. We will see some early works in this direction in chapter 7.
SVC is also able to deal with high dimensional data (see section 3.6) thanks to novel cluster labeling strategies, such as Gradient Descent (GD) and Cone Cluster Labeling (CCL). However, high dimensional datasets can increase the actual SVC execution time, due to the hidden constants in the computation of the distances among vectors.[35] Besides, the SVC is also scalable to very large datasets.
Even though there is no explicit indication of the ability of the SVC to be robust with respect to sparse data and missing values (see section 3.7), we verify this property in our experimental sessions. The work[36] presented in Ancona et al. (2004) suggested us the idea that the SVC could be robust with respect to aforesaid situations.
Finally, if we choose the GD cluster labeling strategy, we get another unique characteristic: the ability of extending clusters to enlarged clustered domains which partition the whole data space in many disjoint subdomains as the number of clusters is. This enable us to decide in what clusters completely unknown points have to be put. For example, if we partition a dataset of only 1000 points in three different clusters, we can successively cluster unknown points, i.e. points originally not in the dataset, by checking those points to which clustered domain belong.
Some drawbacks still remain. First, very high dimensional datasets (thousands and more features) make the clustering process less accurate. Furthermore, the SVC does not fully exploit the SVM characteristics: the whole clustering process is a hybrid process, because only the cluster description stage is a proper SVM application. The cluster assignment stage does not employ the SVM directly, but it exploits the cluster description information to perform the cluster assignment.
By employing a multi-ball description in the feature space is most likely that we can get a better cluster description, especially for strongly overlapping clusters. Some alternative support vector methods for clustering which exploit this idea

[34] Classical clustering algorithms are usually limited to convex shapes such as hyper-ellipsoids.
[35] Generally, this is an issue which affects all clustering algorithms.
[36] Ancona et al. (2004) analyze the influence of data representation on the generalization capacity of an SVM classifier. They have shown that the generalization capacity even increases with a moderated sparseness.

will be reviewed in the next chapter.

6.12 Future work

One of the issues that still remains open is the estimation of the soft margin parameter C through theoretically rooted methodologies, instead of empirical observations. A promising approach to determine better the tradeoff between allowed outliers and accuracy was presented by Yankov et al. (2007).[37]
Much research on cluster labeling could further improve the accuracy and the speed of the whole clustering process. An interesting alternative way for labeling the data is the employment of an SVM classifier for the cluster labeling stage. Early works in this direction was already presented in Ping et al. (2005), Ling and Zhou (2006), and Ling et al. (2006) by applying the method proposed in Jong et al. (2003) after the classic SVC process.
Besides, it would be also interesting to perform tests with alternative SVM formulations (see subsubsection 6.8.1.2) for the cluster description stage.
Finally, it also worths to explore some solutions for exploiting some concepts from the Bregman framework (see chapter 5) in the SVC. See subsection 10.3.1 for more details about this intuition.

[37] See also section 6.9.

CHAPTER

SEVEN

Alternative Support Vector Methods for Clustering

«Evolution is a change from an indefinite, incoherent, homogeneity to a definite, coherent, heterogeneity, through continuous differentiations and integrations.»

HERBERT SPENCER
(SPENCER, 1862)

FOR the sake of completeness in this chapter we briefly review other two remarkable approaches to the clustering via SVMs, already mentioned in the previous chapter: the *Kernel Grower (KG)* (Camastra, 2004, 2005; Camastra and Verri, 2005) and the *Multi-sphere Support Vector Clustering (MSVC)* (Asharaf et al., 2006; Chiang and Hao, 2003).
Both rely on the Support Vector Domain Description (SVDD) process and on the idea of describing the clusters by a dedicated feature space sphere each, but they are significantly different from each other. The first one adopts a *K-means*-like strategy to find spherical-like[1] clusters directly in the feature space. The second solution use an adaptive cell growing model which essentially identifies dense regions in the original space by finding their corresponding spheres with minimal radius in the feature space.
The main difference between them is that the KG algorithm needs the number of clusters in input; on the contrary, MSVC algorithm inherits from the SVC the ability of discovering the clusters structure by itself.

[1]We say "spherical-like" because the SVDD does not always describe a perfect sphere. For $C < 1$ we have spherical-like shapes.

7.1 Kernel Grower

The KG algorithm was first introduced in Camastra (2004). Further published researches about this method seem not to be available, but the author has confirmed that at the moment KG is subject to further research (Camastra, 2007).
To formulate and explain the KG algorithm more easily, let us provide a slightly different formulation (with respect to the formulation in chapter 4) of the K-means algorithm.

7.1.1 K-means alternative formulation

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be the dataset in a d-dimensional space such that $\mathcal{X} = \{\vec{x}_i\}_{i=1,2,\cdots,n}$ where n is the size of the dataset. We call *codebook* the set $\mathcal{W} = \{\vec{w}_i\}_{j=1,2,\cdots,K}$ such that $\mathcal{W} \subseteq \mathcal{X}$ and $K \ll n$. Each element which belongs to the codebook is called *codevector*.

Definition 7.1 (Voronoi Region) *The* **Voronoi Region** *of the codevector $\vec{w}_j \in \mathcal{W}$ is the set of all vectors in $\mathbb{R}^d$ for which $\vec{w}_j$ is the* nearest codevector

$$\mathcal{R}_j = \{\vec{r} \in \mathbb{R}^d \colon j = \arg\min_{1 \leq c \leq K} \|\vec{x} - \vec{w}_c\|\} \tag{7.1}$$

Definition 7.2 (Voronoi Set) *The* **Voronoi Set** *of the codevector $\vec{w}_j \in \mathcal{W}$ is the set of all vectors in $\mathcal{X}$ for which $\vec{w}_j$ is the* nearest codevector

$$\mathcal{V}_j = \{\vec{x} \in \mathcal{X} \colon j = \arg\min_{1 \leq c \leq K} \|\vec{x} - \vec{w}_c\|\} \tag{7.2}$$

Definition 7.3 (Voronoi Tessellation) *The* **Voronoi Tessellation** *is the partition of the $\mathbb{R}^d$ formed by all* Voronoi Regions

In view of the above definitions, we can state that the K-means algorithm works repeatedly moving all codevectors to the arithmetic mean of their Voronoi Sets, until it minimizes the *quantization error*, i.e. the Residual Sum of Squares (RSS) in the case of the K-means (see Equation 4.3).
Therefore, K-means can be summarized in the following steps

(i) Randomly choose K vectors from the dataset $\mathcal{X}$ to initialize the codebook $\mathcal{W}$

(ii) Compute a Voronoi Set for each codevector

(iii) Move each codevector $\vec{w}_j$ to the mean of its Voronoi Set V_j

$$\vec{w}_j = \frac{1}{|V_j|} \sum_{\vec{x} \in V_j} \vec{x} \tag{7.3}$$

(iv) Go back to step (ii) if any codevector has been moved

(v) Return the codebook

The codevectors correspond to the centroids, whereas the Voronoi Sets are the clusters.

7.1.2 Kernel Grower formulation

The KG maps the dataset $\mathcal{X}$ in a high dimensional feature space $\mathcal{F}$ by means of a nonlinear mapping ϕ. Unlike SVC, it considers K different centers in the space $\mathcal{F}$

$$\mathcal{W}_{\mathcal{F}} = \{\vec{a}_i\}_{i=1,2,\cdots,K} \tag{7.4}$$

The $\mathcal{W}_{\mathcal{F}}$ is called *Feature Space Codebook* and the *feature space codevectors* therein are the K aforesaid centers.
By analogy with codevectors in input space, we can define a Voronoi Region and a Voronoi Set for each center.

Definition 7.4 (Feature Space Voronoi Region) *The* **Feature Space Voronoi Region** *of the center $\vec{a}_j \in \mathcal{W}_{\mathcal{F}}$ is the set of all vectors in $\mathcal{F}$ for which $\vec{a}_j$ is the* nearest codevector

$$\mathcal{R}_{\mathcal{F}j} = \{\vec{f} \in \mathcal{F} \colon j = \arg\min_{1 \le c \le K} \|\vec{f} - \vec{a}_c\|\} \tag{7.5}$$

where $\vec{f}$ is any of the points in feature space.

Definition 7.5 (Feature Space Voronoi Set) *The* **Feature Space Voronoi Set** *of the center $\vec{a}_j \in \mathcal{W}_{\mathcal{F}}$ is the set of all vectors in $\mathcal{X}$ such that $\vec{a}_j$ is the* nearest codevector *for their feature space images*

$$\mathcal{V}_{\mathcal{F}j} = \{\vec{x} \in \mathcal{X} \colon j = \arg\min_{1 \le c \le K} \|\phi(\vec{x}) - \vec{a}_c\|\} \tag{7.6}$$

Definition 7.6 (Feature Space Voronoi Tessellation) *The* **Feature Space Voronoi Tessellation** *is the partition of the $\mathcal{F}$ formed by all* Feature Space Voronoi Regions

As already stated, the KG algorithm use a K-means like strategy, i.e. it moves repeatedly all centers in the feature space by computing the SVDD on their Feature Space Voronoi Sets, until no center changes anymore. To make KG robust with respect to the outliers, the SVDD is computed on an ouliter-less subset of the Feature Space Voronoi Sets.
The equation below describes a subset of the Feature Space Voronoi Set $\mathcal{V}_{\mathcal{F}j}$ with potentially no outliers. The paramter ρ can be set up using a model selection technique.

$$\mathcal{V}_{\mathcal{F}j}(\rho) = \{\vec{x} \in \mathcal{V}_{\mathcal{F}j} \colon \|\phi(\vec{x}) - \vec{a}_c\| < \rho\} \tag{7.7}$$

Therefore, KG can be summarized in the following steps

(i) Randomly choose K vectors from the feature space image of dataset $\mathcal{X}$ to initialize the codebook $\mathcal{W}_{\mathcal{F}}$

(ii) Compute the outlier-less version of a Feature Space Voronoi Set, $\mathcal{V}_{\mathcal{F}j}(\rho)$, for each center

Algorithm 15 The Kernel Grower (KG) algorithm

procedure KG($\mathcal{X}, K$)
  $\mathcal{W}_{\mathcal{F}} \leftarrow$ SelectRandomCenters($K, \mathcal{X}$)
  repeat
    for $i \leftarrow 1, K$ **do**
      $\mathcal{V}_{\mathcal{F}j} \leftarrow \{\vec{x} \in \mathcal{X} : j = \arg\min_{1 \leq c \leq K} \|\phi(\vec{x}) - \vec{a}_c\|\}$
      $\mathcal{V}_{\mathcal{F}j}(\rho) \leftarrow \{\vec{x} \in \mathcal{V}_{\mathcal{F}j} : \|\phi(\vec{x}) - \vec{a}_c\| < \rho\}$
    end for
    for $i \leftarrow 1, K$ **do**
      $\vec{a}_j = \text{SVDD}(\mathcal{V}_{\mathcal{F}j}(\rho))$
    end for
  until no center changes
end procedure

(iii) Compute SVDD for all $\mathcal{V}_{\mathcal{F}j}(\rho)$

$$\vec{a}_j = \text{SVDD}(\mathcal{V}_{\mathcal{F}j}(\rho)) \tag{7.8}$$

(iv) Go back to step (ii) if any center has been changed

(v) Return the Feature Space Codebook

The centers correspond to the centroids, whereas the Feature Space Voronoi Sets are the clusters.

7.1.3 Conclusion

The Kernel Grower (KG) algorithm is a kernel-based clustering algorithm which relies on the support vector description of the data, like the SVC. The differences with respect to the latter are immediately evident. The first one is the use of the multi-sphere data description which detect the clusters as spheres in feature space, one for each cluster. As stated in the previous chapter, the multi-sphere approach is a way to tackle highly nonlinear separable problems better. Besides, it is also clear that we do not need anymore a cluster labeling stage, because the whole cluster membership is determined only by means of the Support Vector Domain Description (SVDD).
Since this algorithm adopts a K-means like strategy, it also provides us with some information which the SVC does not provide, such as the *centroids* (or *prototypes*). However, it inherits from K-means some issues too. First, the problem of the estimation of number of clusters was reintroduced. Furthermore, the clustering results highly depend on the initialization of the codebook, so an alternative strategy other than the random one for the selection of the initial codevectors is needed.
Besides, although KG is robust with respect to the outliers, it relies on third party model selection techniques for the estimation of the parameter ρ which rules the outliers handling.

Algorithm 16 The Multi-sphere Support Vector Clustering (MSVC) algorithm

procedure MSVC($\mathcal{X}$)
  $\mathcal{C}_1 \leftarrow \vec{x}_1$ ▷ Initialize first cluster
  $\beta_1 \leftarrow 1$
  $R_1 \leftarrow 0$ ▷ R_i is the radius of the i-th cluster/sphere
  for all $\vec{x} \in \mathcal{X} \setminus \vec{x}_1$ **do**
    if there are clusters available **then**
      $W \leftarrow$ findIndexOfWinningCluster()
      if VigilanceDegree($\vec{x}$) $< \epsilon$ **then**
        $\vec{x} \in \mathcal{C}_W$
        SVDD($\mathcal{C}_W$) ▷ modify the sphere to include new point
        enable all disabled clusters
      else
        disable $\mathcal{C}_W$ by setting $d(\vec{x}, \mathcal{C}_W)$
      end if
    else
      $\mathcal{C}_j \leftarrow \vec{x}$ ▷ create a new cluster
      $\beta_{\vec{x}} \leftarrow 1$
      $R_j \leftarrow 0$
      enable all disabled clusters
    end if
  end for
end procedure

From a time complexity point of view, the computation of several MEB (one for each cluster) for a number of iterations can be considered a drawback. Anyway, alternative SVM formulations such as CVM or optimization algorithms such as SMO make this problem less onerous.

7.2 Multi-sphere Support Vector Clustering

The Multi-sphere Support Vector Clustering (MSVC) was initially introduce by Chiang and Hao (2003) and is the second attempt of exploiting the idea of the multi-sphere cluster computation in feature space. This peculiarity makes the MSVC the second support vector method for clustering that use only SVMs to perform the clustering, i.e. no cluster labeling stage anymore.
Moreover, the multi-sphere structure implies that we can obtain the *centroids* also in this case. Once we have found the spheres, their centers correspond to centroids.

7.2.1 Process overview

Let us provide an overview on the whole clustering process performed by the MSVC in order to understand better the details in the sequel.

The MSVC algorithm starts by initializing the first cluster with just one point. Then, it performs the *winning cluster competition* stage for finding (among the clusters already built) a suitable cluster for a given input point. Once it has found a winning cluster, it performs a *validity test* to verify if the current point can be really put into the cluster found. If the validity test gives a positive result, then the point is put into the winning cluster, otherwise the cluster found is disabled and the algorithm looks for another winning cluster. The new search does not take into account the disabled clusters. If no other clusters are available, a new cluster is created and initialized with the current point.
The whole process is repeated until all points are clustered.

7.2.2 Process in details

The whole clustering process performed by the MSVC involve five subprocesses: *Initialization*, *Winning Cluster Competition*, *Validation Test*, *Learning Cluster Parameters* and *New Cluster Creation*. All these procedures are combined to obtain the algorithm whose pseudocode we can see in Algorithm 16.
The hyper-parameters of the MSVC are all the ones inherited from the SVC with another one in addition. Therefore, we still have the kernel width q and the soft margin parameter C. The new hyper-parameter is the *vigilance threshold* ϵ and its value determines how close an input point must be to a cluster in order to match. In other words, it determines whether the algorithm creates a new cluster for the current point or modifies the winning cluster to enclose such a point.
In the following subsections all stages of the process are presented. Expect the initialization, all other stages are repeated for all point in the input dataset.

7.2.2.1 Initialization

The initialization process set up the first cluster with the first point. Moreover, the Lagrangian multiplier of such a point is set to 1 and the initial radius of the sphere-cluster is set to zero.

7.2.2.2 Winning cluster competition

For each input point $\vec{x}$, the nearest candidate cluster was considered as the winning cluster, i.e. the cluster which could enclose the point $\vec{x}$ depending on the parameter ϵ. To decide which cluster is closer to the point, we have to compute a distance between the point and the clusters. There are two possible ways to compute this distance: the first way is to compute the distance in feature space, the alternative way is to compute the distance in data space.
Let $\vec{a}$ be the center of a sphere-cluster $\mathcal{C}$ in feature space. We can compute the distance in feature space $\mathcal{F}$ as follows (see Equation 6.15)

$$d_{\mathcal{F}}(\vec{x}, \mathcal{C}) = \|\phi(\vec{x}) - \vec{a}\|^2 = K(\vec{x}, \vec{x}) - 2\sum_{k=1}^{n_{sv}} \beta_k K(\vec{x}_k, \vec{x}) + \sum_{k=1}^{n_{sv}} \sum_{l=1}^{n_{sv}} \beta_k \beta_l K(\vec{x}_k, \vec{x}_l) \quad (7.9)$$

In data space the distance looks as follows

$$d_{\mathcal{X}}(\vec{x}, \mathcal{C}) = \frac{1}{|\mathcal{C}|} \sum_{\vec{y} \in \mathcal{C}} \|\vec{x} - \vec{y}\|^2 \tag{7.10}$$

Both distances are eligible since the distance in the feature space is proportional to the distance in the input space.

7.2.2.3 Validation test

In order to verify the validity of the winning cluster, the so-called *cluster vigilance test* is then conducted. The vigilance function[2] $g_W(\vec{x})$ defines the so-called vigilance degree that $\vec{x}$ belongs to the winning cluster $\mathcal{C}_W$. The point match the cluster $\mathcal{C}_W$ if and only if $g_W(\vec{x}) < \epsilon$. Then, the algorithm enters the *Learning Cluster Parameters* stage and modifies the cluster $\mathcal{C}_W$ to enclose the point $\vec{x}$. If the vigilance degree is greater than the vigilance threshold, then the algorithm disable the cluster $\mathcal{C}_W$, i.e. the cluster $\mathcal{C}_W$ will be no available for the next *Winning Cluster Competition* stage. If no other cluster is available (e.g. all clusters has been disabled), the algorithm enters the *New Cluster Creation* stage.

7.2.2.4 Learning cluster parameters

This stage is the most time-consuming stage of the whole algorithm, because it consists of solving an instance of the SVDD problem. In fact, in this stage the radius and the center of a sphere-cluster are modified in order to enclose the matched point in addition to the former ones. This generates a new set of support vectors and corresponding Lagrange multipliers.

7.2.2.5 New cluster creation

When no winning clusters were found, the current point is supposed to belong to another, new, cluster. This stage is very similar to the initialization stage. It sets up the k-th cluster with the current point and the Lagrangian multiplier of such a point is set to 1. Besides, the initial radius of the sphere-cluster is set to zero.

7.2.3 Conclusion

The MSVC seems to be a promising approach to clustering by means of the support vector machines. It relies on multi-sphere approach, so strongly overlapping clusters are handled well. Moreover, it inherits most of the advantages of the SVC, such as the arbitrariness of the cluster shapes, the robustness with respect to the outliers, and the ability to discover the number of clusters. Thanks to the multi-sphere approach, it also provides informations about centroids alongside the support vector description of the data. Furthermore, a soft (or fuzzy) clustering interpretation was also provided in Chiang and Hao (2003) and in Zheng et al. (2006).

[2] More details on the vigilance function are in Chiang and Hao (2003, sec. 4).

The major drawback of this approach is the introduction of a third parameter, the vigilance threshold ϵ. It makes the overall parameters tuning harder. Besides, the previous works for the estimation of the kernel width q is not applicable in this case.

7.3 Overall conclusion

In this chapter we reviewed some alternative support vector methods for clustering, which use the multi-sphere approach both. This is the main difference with respect to the SVC presented in chapter 6, because this makes the KG and the MSVC able to perform the whole clustering process only by means of the SVMs. Thanks to the multi-sphere approach, both algorithms presented here are able to provide cluster centroids in addition to the support vector data description.
At the present stage, these works are very experimental and further researches are needed to understand the future applicability spectrum. Anyway, both works sounds very interesting.

CHAPTER

EIGHT

Support Vector Clustering software development

«It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration.»

EDSGER WYBE DIJKSTRA
(DIJKSTRA, 1982, PP. 129-131)

SINCE the Support Vector Clustering (SVC) is a very experimental technique, no affirmed software was available to perform experiments. An embryonal version of software written in C++ is available online[1] thanks to Nath and Shevade (2006). However the software is definitely not modular and not documented, so it is very difficult to adapt it for the own need. Besides, it uses a preprocessing step we do not want to use and, for aforementioned reasons, it is too hard to remove it and use a simple SVC scheme. The other piece of software available is a MATLAB toolbox.[2] For the same reasons, the software is useless in our case.

Therefore, we decided to start the development of a new software[3] for the SVC that was modular, reliable and extensible. Due to these requirements we choose an Object Oriented Programming (OOP) approach. For the sake of efficiency, the

[1] See `http://saketh.nath.googlepages.com/research`.

[2] It was sent to us by Lee and Lee (2005).

[3] A copy of the current version of the software is available at `http://thesis.neminis.org/2007/12/03/support-vector-clustering-code/`.

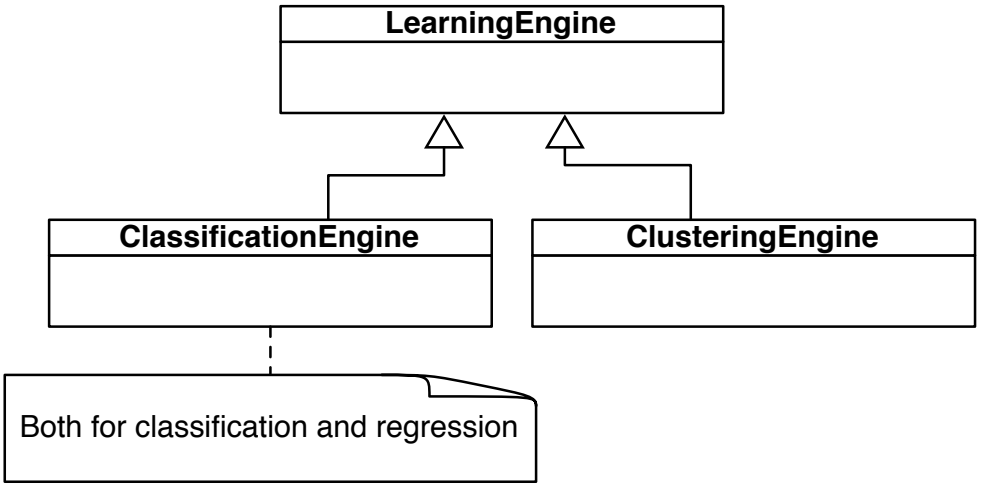


Figure 8.1: *The first three basic classes of the* damina *library. They are all interfaces which establish the mandatory methods for all implementing classes.*

programming language chosen is the C++. We refer to this new software library with the code-name *damina*.[4]

8.1 Software foundations

The first rule in software development is the reutilization of the code. This is the reason why software libraries exist. To avoid the development of yet coded software, we have first performed a research to find software libraries which were suitable for our work. The results of this research are our software foundations.

8.1.1 Software for the cluster description stage

The cluster description stage of the SVC (see subsection 6.1.1) is based on the SVDD. However, the only software available for the SVDD is the *DD_tools*, a MATLAB toolbox (Tax, 2007). Further researches have revealed that there is another SVM formalization for the *One class classification* problem: the *One class SVM* by Schölkopf et al. (2000b), that is equivalent to SVDD under certain conditions (see Appendix A).
Thanks to such an equivalence, we can employ the LIBSVM (Chang and Lin, 2007) as part of the basics for our software development. LIBSVM is a C++ library which provides the implementation for a variety of SVM problems, including the the *One class SVM*. In LIBSVM the Quadratic Programming (QP) problem for the training of an SVM is solved thanks a recent SMO-type algorithm (Fan et al., 2005). Moreover, LIBSVM also provides the n-fold cross-validation, the shrinking heuristics by Joachims (1998) and a cache for the kernel matrix.

[4] *Da*ta *mi*ning *Na*ples.

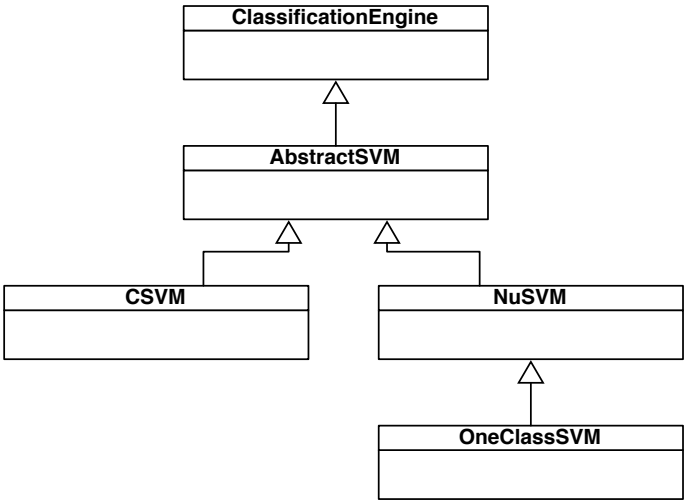


Figure 8.2: *The* AbstractSVM *class inherits from the* ClassificationEngine *and provides all the basic methods for all SVMs. The* OneClassSVM *is not a direct child of the* AbstractSVM*: it inherits from the* NuSVM*, which implements the ν-SVM formalism. The* CSVM *implements the classic Vapnik SVM formulation.*

8.1.2 Software for connected components

Since the connected components of a graph is a recurrent problem in all cluster labeling algorithms presented in section 6.3, we decided to employ an efficient and stable software library: the *Boost Graph Library*, that is part of the *Boost C++ Libraries* project (Various, 2007).[5] In order to ensure efficiency and flexibility, Boost makes extensive use of templates. Boost has been a source of extensive work and research into generic programming and metaprogramming in C++.

8.2 Class structure

Our initial goal was to develop a software which was focused only on the support vector clustering. Later, we have opened the roadmap of the project out due to the interest of the VO-Neural[6] team in our work. The VO-Neural project aims at developing a data mining library which includes several solutions for classification, regression, clustering and so on.
In view of the above events, the VO-Neural team and us have agreed up on a preliminary class structure for the design of the software (see Figure 8.1 and Figure 8.2).
The root of the whole class hierarchy is the *LearningEngine* that plays the role of a

[5]Many of Boost's founders are on the C++ standard committee. Visit also `http://www.boost.org`.

[6]The VO-Neural project, formerly known as Astroneural, is a project of the Physics department at the University "Federico II" of Naples, Italy. For more information, visit `http://people.na.infn.it/~astroneural/`.

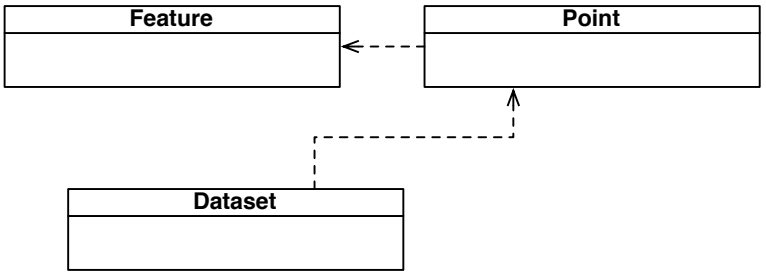


Figure 8.3: *A* Dataset *is a collection of* Point *which, in turn, is a collection of* Feature*.*

generic *learner*. It provides only the basic interface for any type of learner. *ClassificationEngine* and *ClusteringEngine* are direct children of the *LearningEngine*. *ClassificationEngine* provides the interface for implementing a classifier or a regressor.[7] Instead, *ClusteringEngine* provides the interface to built a clustering algorithm.
These are the most generic classes of the *damina* project. Then, there are a number of classes related to the Support Vector Machines (SVMs). First, the *AbstractSVM* class which inherits from *ClassificationEngine* and provides the basic behaviors for a Support Vector Machine (SVM). In this case exposes all the basic functionalities that LIBSVM places at our disposal, such as the choice of the kernel, the setting of the hyper-parameters, the setting of other parameters (tolerance, kernel cache size, etc.), the loading of the training and test sets, the control over the cross-validation and the shrinking heuristics (Joachims, 1998).
The direct children of the *AbstractSVM* class are the *CSVM* and the *NuSVM*. The first one implements the specific behavior of the classic Vapnik SVM formulation. The second one implements the specific characteristics of the ν-SVM formalism by Schölkopf et al. (2000a). The *NuSVM* class is also the base class for the *OneClassSVM* class. *CSVM*, *NuSVM* and *OneClassSVM* are all built as wrappers of LIBSVM functions.
The *OneClassSVM* provides us with the instruments for performing the cluster description stage of the support vector clustering.

8.2.1 Datasets

As far as the datasets are concerned, we have developed a set of classes to manage both the training sets and the test sets. We designed three simple classes: the *Feature* class encapsulates the "feature" concept. Each feature is treated as a index-value couple, where the "index" is the original position of the feature as component of the related vector. Then, we have the *Point* class which is a collection of features, i.e. we have a sparse representation of the data. Moreover, the *Point* class have an attribute "label". In the "label" attribute is stored the class (if available) which a point belongs to. Finally we have designed the *Dataset* class

[7] A classifier predicts a categorical attribute, while a regressor predicts a numerical attribute. However, in the implementation the "all is a number" rule holds, so both supervised learners can be represented by the same software module.

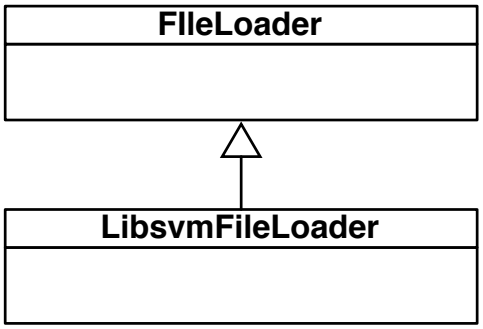


Figure 8.4: *The* FileLoader *interface and the* LibsvmFileLoader *class.*

as a collection of points. This data modeling reflects the data structures of the LIBSVM, that efficiently avoid to store the zero values. Moreover, the *Dataset* class has also an attribute called "labels". The "labels" attribute is an array whose size is the same of the dataset. Such an array stores the cluster assignment of the points when the dataset is involved in a clustering problem.

8.2.2 File loaders

To transparently deal with different file formats we have introduced the "file loader" concept. An interface class, *FileLoader*, provides the guidelines for the implementation of a file loader. Currently, there is only a concrete file loader, the *LibsvmFileLoader*. Such a loader is able to load datasets in the LIBSVM format and is also able to convert the data into the internal *damina* representation (see subsection 8.2.1).
The "file loader" hierarchy allows to easily adds the support to new file formats.

8.3 Support Vector Clustering

The original formulation of the SVC algorithm is implemented into the *SVClustering* class. It uses the *OneClassSVM* class for the cluster description and by means of the Boost Graph Library implements the method to perform the cluster labeling stage as originally formulated by Ben-Hur et al. (2001). It also provides the methods for the calculation of the sphere radius and the distance of a point image from the center of the sphere. For its internal utilization (and of its children classes), there are some protected methods: for retrieving Lagrangian multipliers, the clustering results, the number of clusters, the number of valid clusters (with the cardinality greater than a prefixed threshold), etc.
The *CCLSVClustering* class extends the *SVClustering* class. This new class simply overrides the cluster labeling stage with our version of the Cone Cluster Labeling (CCL) algorithm.[8]

[8]It assigns BSV points to clusters at the end of the process, by measuring the proximity with

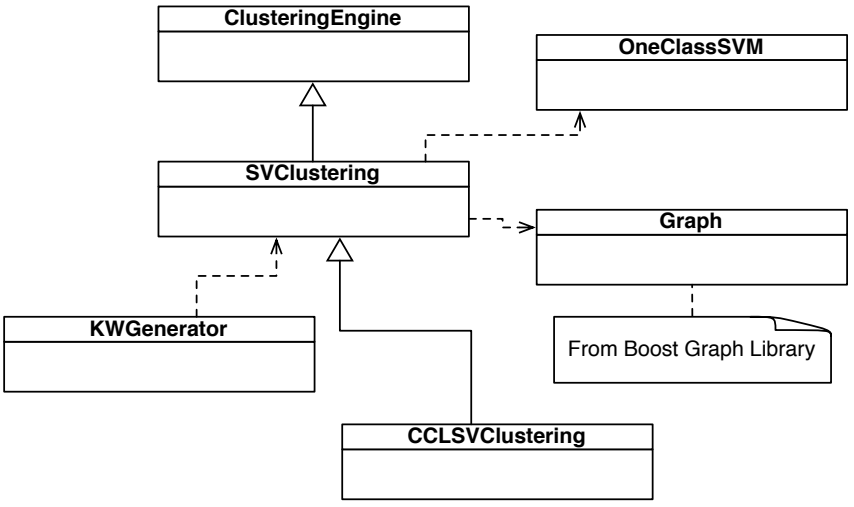


Figure 8.5: *The* SVClustering *inherits from* ClusteringEngine *and implements the original Support Vector Clustering (SVC) algorithm. It uses also the* OneClassSVM *class and the* Graph *one from the Boost library. The* CCLSVClustering *inherits from* SVClustering *and overrides the cluster labeling algorithm.*

Finally, the secant-like kernel width generator with our contributions (see subsection 6.4.1) is implemented in the class *KWGenerator*. Our implementation of the secant-like kernel width generator reflects our idea of reducing the time complexity, therefore it directly accesses to the informations stored into an instance of the *SVClustering*, in order to compute the data needed to estimate the next kernel width value.
As stopping criteria we use a combination of some criteria discussed in section 6.6: the threshold on the number of support vectors, the slope of the last generated secant line, the number of clusters.

8.4 Companion functions

It is common to perform a number of clustering experiments on labeled dataset in the research environment. Thanks to the LIBSVM file format we easily included in *damina* project the automatic calculation of clustering quality via the external criteria discussed in subsection 3.3.2. In fact, the LIBSVM file format allows to specify a class label for each item. This information is usually exploited by supervised learning algorithm like classifiers and regressors, whereas they are ignored by unsupervised learning algorithm. In our case we use class labels information for measuring the quality of the clustering by means of external criteria and easing the end-user job.

respect to all points instead to SV points only (see subsection 6.3.5).

8.5 Conclusion

In this chapter we provided an overview on the development of a new software, code-named *damina*. Such a software relies on robust software basics, such as the LIBSVM and the Boost C++ Libraries. The *damina* project is currently in alpha status and is far to be stable and complete. Anyway, it includes a first implementation of the state-of-the-art of the support vector clustering researches, such as the Cone Cluster Labeling (CCL) and the secant-like kernel width generator. So, we also provide a practical contribution to the SVC.

8.6 Future work

For the sake of accuracy and in order to perform more robust comparisons with other clustering algorithms, an improved and extended software for the Support Vector Clustering (SVC) is needed. More stability and reliability is needed. Moreover, it is important to implement all the key contributions to this promising technique proposed all around the world. Such a complete software would prove better the quality of this clustering algorithm. In fact, all the tests have been currently performed by exploiting only some characteristic and/or special contribution per time.
In the future we expect to implement all available clustering labeling algorithms (see section 6.3), the *kernel whitening* to improve the cluster description (see subsubsection 6.1.1.2), the SVC validity measure (see subsubsection 6.6.3.1) and the stopping criterion based on validity indices (see subsection 6.6.3), the estimation of the tradeoff between outliers percentage and accuracy (see section 6.9), and all other main contributions to the SVC. In addition, we expect to implement other support vector methods for clustering too, in order to perform comparative experiments among these fashionable clustering methods.

CHAPTER

NINE

Experiments

«It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong.»

RICHARD PHILLIPS FEYNMAN
(UNSOURCED)

EXPERIMENTAL results are the keystone for good theories. Our experimental session is organized in several stages. The first stage aims at verifying the basic capabilities of both techniques introduced in the previous chapters as well as to underline the basic differences between them. At the same time, we also verify how these two clustering techniques deal with different application domains.
The goal of the second stage is to prove the outliers handling ability of the SVC with respect to the Bregman co-clustering (and, generally, all the K-means-like clustering algorithms) by means of some tests performed on synthetic datasets specifically built for this stage.
The third stage is a collection of experiments about a specific application domain: the astrophysical data. Within the limits of this application domain we perform two different kind of tests. The first type of experiments aims at verifying the ability of the two clustering techniques to cluster data with missing values. The latter is one of the most important problems in the astrophysics and other scientific fields, as already anticipated in section 3.7. The second type of problem we faced within the astrophysics domain is the unsupervised separation of the stars from galaxies. The star/galaxy separation is a very hard problem from an unsupervised learning viewpoint, so we exploited it to verify the differences between the two techniques when the problem at hand is heavily nonlinear separable.

Finally, we faced the clustering of textual documents, to discover which technique fit this application domain best and also for verifying the ability of the two clustering algorithms to deal with very high dimensional data.[1] The theory about the two clustering techniques discussed in chapter 5 and chapter 6 suggests that the Bregman co-clustering should perform the best. Since data matrices that represent text documents are sparse, the experiments about these data will also provide us with results about the robustness of the algorithms with respect to the data sparseness (see section 3.7).
In the sequel we first introduce the experimental environment, i.e. hardware, operating system, software. We first present the stage we called *preliminary experimental stage*, where we test on some well-known datasets in the machine learning literature. The original goal of these tests was to verify the applicability of the two techniques to a variety of problems. Anyway, the results also provided some informations about the peculiarities of each technique.
The experiments review follows with the verification of the outliers handling ability, the missing values robustness on astronomical data, strongly overlapping clusters handling ability on star/galaxy separation problem. Finally, we discuss text mining experiments that also give us important informations about the high dimensional data handling ability of both techniques.

9.1 The experimental environment

This section provides an overview on our experimental environment, describing the hardware and the software characteristics of the "brute force" that helped us to accomplish our testing phase.

9.1.1 Hardware and Operating System

The machine used for the tests is an Apple computer with the following configuration

- Machine Name: Power Mac G5
- Machine Model: PowerMac7,3
- CPU Type: PowerPC G5 (3.0)
- Number Of CPUs: 2
- CPU Speed: 2 GHz
- L2 Cache (per CPU): 512 KB
- Memory: 2.5 GB
- Bus Speed: 1 GHz

[1] Usually, a text document in vector space model is represented by thousands, even millions of features (e.g. the NewsGroup20 (NG20) version used by Keerthi and DeCoste (2005), available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#news20.binary`).

and the operating system is

- Product Name: Mac OS X Sever
- Product Version: 10.4.10
- Kernel Version: Darwin 8.10.0

9.1.2 The software

The software for the Support Vector Clustering (SVC) is an alpha version of our project presented in chapter 8. Our SVC software is at a very early stage and it only implements[2] the Cone Cluster Labeling (CCL) as cluster assignment algorithm. However, it resulted reliable.
To test the Bregman co-clustering there are two softwares available. The first one is the *Co-cluster* software developed by Cho et al. (2004b). Since this software was developed before the idea of the general Bregman framework presented in chapter 5, it only implements the instances in Dhillon et al. (2003b) and Cho et al. (2004a), i.e. the scheme $\mathcal{C}_5$ for the I-divergence and the schemes $\mathcal{C}_2$ and $\mathcal{C}_6$ for the Euclidean distance. This software is quite stable, reliable and complete: it also implements different strategies for the co-clustering initialization and the row/column clusters update.
An early implementation of the more general Bregman Co-clustering framework was provided in the software *Bregman Co-cluster*, developed[3] for the work in Huynh and Vu (2007). This software implements all six co-clustering bases both for the Euclidean distance and for the I-divergence. Although this software is less reliable on the average, and has less accessory features, it behaves well in most of the cases, so we used it.[4]

9.2 General setup of the algorithms

The general setup of the two algorithms is always the same for all experiments.[5] As far as the Bregman co-clustering is concerned, each approximation scheme was run twenty times, the initialization of the co-clustering was random and the updating scheme of the row and column clusters is the one described in chapter 5. Initialization strategies other than the random one and alternative updating schemes are available in the software used here,[6] but our setup generally produce very good results for the data tested here. The employment of more enhanced

[2]Actually, it also implements the original Complete Graph (CG) cluster labeling algorithm, but it does not perform well in most of cases, so we do not take it into account.

[3]A copy of the software is currently available at `http://www.cs.utexas.edu/~hntuyen/projects/dm/`.

[4]Only for the last two text clustering experiments we used the *Co-cluster* software.

[5]Unless differently specified

[6]The *Bregman Co-cluster*, that implements all approximation schemes for Euclidean distance and I-divergence.

strategies[7] for cluster initialization and update would only increase the running time.
The SVC probes the data at different levels of details thanks to the secant-like kernel width generator. Since the data used here are labeled, one of the stopping criteria used is the one described in subsection 6.6.1. The only other stopping criterion used is the originally proposed criterion based on the number of support vectors (see section 6.6); such a criterion is used to avoid the software cycling forever due to data that it cannot split.

9.3 Evaluation and comparison of clustering results

Since the data used over the whole experimental phase are labeled, we used *external criteria* (see subsection 3.3.2) to evaluate the clustering results. Generally, we employed the *Accuracy* and the *Precision/Recall/F1* measures. When we dealt with multi-class problems, we also used the *Macroaveraging* measure (also written as Macro-AVG), that is the arithmetic mean of the F1 measures calculated for each class.[8]
Employing different external criteria was more intuitive in some kind of experiments, such as the *Completeness* and the *Contamination* for problems like the benign/malignat cancer separation or the star/galaxy separation. Anyway, the external criteria used for each experiment will be always clear from the context.

9.3.1 Comparing results

In the first place, the results of the Bregman Co-clustering and the Support Vector Clustering were compared to each other in every test. However, a comparison with clustering results obtained by other clustering algorithms is very significant. In some experiments we were able to compare our results with results obtained by other researchers about the same data, either using a different clustering algorithm or the same algorithm in different circumstances.[9] In experiments where no previous work results were available, we were always able to compare our specific results with results obtained by the classical K-means algorithm reproduced by means of the basis[10] $\mathcal{C}_2$ of the Bregman framework without feature clustering enabled.

[7]For example, the *local search* strategy for co-cluster update (Cho et al., 2004a; Dhillon and Guan, 2003a), or the *spectral initialization* of clusters (Cho et al., 2004a).

[8]Since we did an in-software computation of the Macroaveraging measure before truncating the F1 values, it is likely that calculating the macroaveraging by means of the F1 values reported in this chapter the reader does not obtain the same result. The same principle holds for other similar situations.

[9]Different algorithm implementation, different data preprocessing, and so on.

[10]We recall that the notation $\mathcal{C}_i$, $i = 1, 2, \cdots, 6$, refers to one of the six approximation schemes of the Bregman co-clustering (see subsection 5.4.3).

9.4 Preliminary experiments: the data

The preliminary test session aims at verifying the basic capabilities of both algorithms as well as at underlining the basic differences between them. The original goal of these tests was to verify the applicability of the two techniques to a variety of application domains. Anyway, the results also provided some informations about the peculiarities of each technique.
We first present the datasets[11] used in this stage of the experiments and then, in the next section, we present the clustering results of both the Bregman Coclustering and the Support Vector Clustering.

9.4.1 Iris Plant Database

This is perhaps the best known database in literature (Fisher, 1936). The data set contains three classes of fifty instances each, where each class refers to a type of iris plant: *Iris Setosa*, *Iris Versicolour*, *Iris Virginica*. One class is linearly separable from the other two; the latter are not linearly separable from each other. Each item is represented by four attributes: sepal length, sepal width, petal length, petal width. All attributes are expressed in centimeters.

9.4.2 Wisconsin Breast Cancer Database

This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg (Mangasarian et al., 1990). The dataset contains 699 instances represented by 10 attributes. Each instance is in one of two possible classes: benign or malignant. The attributes are: *Clump Thickness*, *Uniformity of Cell Size*, *Uniformity of Cell Shape*, *Marginal Adhesion*, *Single Epithelial Cell Size*, *Bare Nuclei*, *Bland Chromatin*, *Normal Nucleoli*, *Mitoses*. The number of benign instances is 458, whereas the number of malignant ones is 241. There are 16 instances that contain a single missing (i.e., unavailable) attribute value.

9.4.3 Wine Recognition Database

These data were originally used in Forina et al. (1988). They are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivations. The analysis determined the quantities of thirteen[12] continuos attributes, i.e. thirteen constituents found in each of the three types of wines. There are 178 items: 59 in the first class, 71 in the second class and 48 in third class.

[11] Unless differently specified, the following datasets are obtained from the *UCI Machine Learning Repository* (Asuncion and Newman, 2007).

[12] The original 30-dimensional version of dataset is not available anymore.

Dataset	# of items (n)	# dimensions (d)	# of classes
Iris Plant Database	150	4	3
W. Breast Cancer Database	699	10	2
Wine Recognition Database	178	13	3
Quadruped Mammals	500000	72	4
Mushroom Database	8124	112	2
Internet advertisement	3279	1558	2

Table 9.1: *Summary of datasets used in the preliminary stage of experiments.*

9.4.4 Quadruped Mammals

The *Quadruped Animals Data Generator* is a data generator of structured instances representing quadruped animals as used by Gennari et al. (1989) to evaluate the CLASSIT unsupervised learning algorithm. Instances have 8 components: neck, four legs, torso, head, and tail. Each component is represented as a simplified/generalized cylinder. Each cylinder is itself described by 9 attributes: three attributes for location, three for axes, and one attribute for height, radius, and texture each. The program generates instances of the four classes with the following probability: 25.46% for the *giraffe* class, 25.50% for the *dog* class, 24.68% for the *cat* class and 24.36% for the horse class.
Each class has a prototype; the prototype of the selected class is perturbed according to a distribution described in the code for the four classes (i.e., parameterized means with Guassian distributions are used to represent prototypes and perturbation distributions, where the means are used to distinguish the four classes). Each generated item has 72 attributes.
Due to the incompatibility of the generator with our operative environment, we fell back on the ready-to-use dataset[13] used in Orlandic et al. (2005), that has 500,000 normalized instances.
Anyway, the Quadruped Animals Data Generator is available at *UCI Machine Learning Repository* (Asuncion and Newman, 2007).

9.4.5 Mushroom Database

Mushroom records drawn from *The Audubon Society Field Guide to North American Mushrooms*. This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. The latter was combined with the poisonous class. The dataset contains 8124 points represented by 22 nominal attributes.
Since the softwares we used cannot handle nominal attributes, we employed a different version of this dataset, obtained from the LIBSVM Data Repository.[14] Each nominal attribute in the original version was expanded into several binary

[13] Currently, a copy of such a dataset is available at `http://uisacad2.uis.edu/dstar/data/clusteringdata.html`.

[14] Visit `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

attributes and the original 12th attribute was not used because it had missing values. Therefore, the number of features of this new version is 112. The number of edible mushrooms is 3916.

9.4.6 Internet advertisements

This dataset represents a set of possible advertisements on Internet pages (Kushmerick, 1999). The features encode the geometry of the image (if available) as well as the phrases occurring in the URL, the image's URL, the text of the "alt" attribute of the "img" HTML tag, the anchor text, and the words occurring near the anchor text. The task is to predict whether an image is an advertisement or not. The datasets have 3279 instances: only 458 are advertisements. Each item is represented by 1558 attributes. The first three encode the geometry and are continuos. The remaining attributes are binary. There is also 28% of instances that have missing values in some of the three continuos attributes.

9.5 Preliminary experiments: the results

In this section we are going to present the results obtained on the datasets summarized in the previous subsection.
We show some specific abilities of both algorithms. We underline the Bregman co-clustering ability of improving results thanks to the compression along the columns (the features) of the data matrix. In addition, we highlight how some our contributions to the SVC helped us to obtain better results with respect to the "basic" version of the algorithm.
Finally, for this preliminary experimental stage we also provide the CPU and memory usage for each algorithm in order to figure out the different computational characteristics of the two algorithms. These informations are available in Appendix B.

9.5.1 Iris Plant Database

These data are not linearly separable, so we had the first chance to verify the ability of the SVC to deal with nonlinear separable problems as it is a kernel method and also to prove the effectiveness of our SVC enhancements. On the other hand we were also able to verify whether the employment of loss functions other than the Euclidean one and the compression along the columns performed by the Bregman co-clustering provide some benefits in this type of problems.
Finally, a comparison with results found in Camastra (2005) revealed that the algorithms used here outperform both classical algorithms (*SOMs*, *K-means*, *Neural Gas*, *Ng-Jordan Algorithm*) and experimental ones (Kernel Grower (KG)).[15]

[15]For more information about the aforesaid algorithms, see Camastra (2005) and references therein.

Iris Plant Database - Euclidean - No feature clustering									
Basis	Setosa			Versicolour			Virginica		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_{1,2,4}$	92.593%	100%	96.154%	74.138%	86%	79.63%	92.105%	70%	79.546%
$\mathcal{C}_3$	42.553%	40%	41.238%	43.182%	38%	40.426%	47.458%	56%	51.376%
$\mathcal{C}_5$	52.632%	40%	45.454%	21.212%	14%	16.868%	43.038%	68%	52.714%
$\mathcal{C}_6$	**100**%	**100**%	**100**%	**78.947**%	**90**%	**84.112**%	**88.372**%	**76**%	**81.72**%
Iris Plant Database - Information-Theoretic - No feature clustering									
Basis	Setosa			Versicolour			Virginica		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_{1,2,4}$	92.593%	100%	96.154%	74.138%	86%	79.63%	92.105%	70%	79.546%
$\mathcal{C}_3$	27.083%	26%	26.53%	29.032%	36%	32.142%	32.5%	26%	28.888%
$\mathcal{C}_5$	41.509%	44%	42.718%	40.323%	50%	44.644%	37.143%	26%	30.588%
$\mathcal{C}_6$	**100**%	**100**%	**100**%	**93.878**%	**92**%	**92.93**%	**92.157**%	**94**%	**93.07**%

Iris Plant Database - Euclidean - No feature clustering				
	$\mathcal{C}_{1,2,4}$	$\mathcal{C}_3$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	85.333%	44.667%	40.667%	**88.667%**
Macroaveraging	85.11%	38.345%	29.213%	**88.611%**
Iris Plant Database - Information-Theoretic - No feature clustering				
	$\mathcal{C}_{1,2,4}$	$\mathcal{C}_3$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	85.333%	29.333%	40%	**95.333%**
Macroaveraging	85.11%	29.187%	39.317%	**95.333%**

Table 9.2: *Bregman Co-clustering of the Iris Plant Database, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. The $\mathcal{C}_6$ scheme yields the best results both with Euclidean Co-clustering and Information-Theoretic Co-clustering. The best results are highlighted in bold.*

9.5.1.1 Bregman Co-clustering

In Table 9.2 we can see the results obtained by the Euclidean Co-clustering and the Information-Theoretic Co-clustering without feature clustering, i.e. both are used in a classical one-way manner. The performance of the $\mathcal{C}_3$ and $\mathcal{C}_5$ schemes are very poor for both divergences. The remaining schemes for the Euclidean distance have performance very close to the results already obtained with the K-means in the past (Camastra, 2005, Table 1). On the contrary, we can see that the I-divergence loss function performs well on Iris data and outperforms the Euclidean distance. However, in both cases the best scheme is $\mathcal{C}_6$.
Watching at Table 9.3 we can note that the feature clustering[16] further improves the results obtained without feature clustering. The first thing we see is that the feature clustering has significantly improved the schemes $\mathcal{C}_3$ and $\mathcal{C}_5$. Moreover, the scheme $\mathcal{C}_5$ with the I-divergence gives the best results on these data. However, also other schemes have been improved.
With or without feature clustering, a connection among the schemes $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_4$ is evident. Without feature clustering the three schemes perform the same

[16] We requested two feature clusters here.

Iris Plant Database - Euclidean - With feature clustering									
Basis	Setosa			Versicolour			Virginica		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_{1,2}$	92.593%	100%	96.154%	74.138%	86%	79.63%	92.105%	70%	79.546%
$\mathcal{C}_3$	100%	100%	100%	67.273%	74%	70.476%	71.111%	64%	67.368%
$\mathcal{C}_4$	98.039%	100%	99.01%	78.333%	94%	85.454%	94.872%	74%	83.146%
$\mathcal{C}_5$	**100**%	**100**%	**100**%	**85.714**%	**96**%	**90.566**%	**95.455**%	**84**%	**89.362**%
$\mathcal{C}_6$	100%	100%	100%	92.857%	78%	84.782%	81.034%	94%	87.036%
Iris Plant Database - Information-Theoretic - With feature clustering									
Basis	Setosa			Versicolour			Virginica		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_1$	92.593%	100%	96.154%	74.138%	86%	79.63%	92.105%	70%	79.546%
$\mathcal{C}_{2,4}$	100%	100%	100%	95.455%	84%	89.362%	85.714%	96%	90.566%
$\mathcal{C}_3$	100%	98%	98.99%	88.889%	80%	84.21%	82.143%	92%	86.792%
$\mathcal{C}_5$	**100**%	**100**%	**100**%	**100**%	**90**%	**94.736**%	**90.909**%	**100**%	**95.238**%
$\mathcal{C}_6$	100%	100%	100%	91.837%	90%	90.91%	90.196%	92%	91.09%

Iris Plant Database - Euclidean - With feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	85.333%	85.333%	79.333%	89.333%	**93.333%**	90.667%
Macroaveraging	85.11%	85.11%	79.281%	89.203%	**93.309%**	90.606%
Iris Plant Database - Information-Theoretic - With feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	85.333%	93.333%	90%	93.333%	**96.667%**	94%
Macroaveraging	85.11%	93.309%	89.997%	93.309%	**96.658%**	94%

Table 9.3: *Bregman Co-clustering of the Iris Plant Database, with two feature clusters requested. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. The $\mathcal{C}_5$ scheme yields the best results both with Euclidean Co-clustering and Information-Theoretic Co-clustering. The best results are highlighted in bold.*

both with Euclidean distance and with I-divergence. When we apply the feature clustering, we see a link between the scheme $\mathcal{C}_1$ and $\mathcal{C}_2$ in Euclidean Co-clustering, whereas in Information-Theoretic Co-clustering the link between the scheme $\mathcal{C}_2$ and $\mathcal{C}_4$ still holds.

Anyway, the connection among the schemes is not always the same for every dataset, but it is interesting to note that for a particular dataset there are some approximation schemes that yield the same results.

9.5.1.2 Support Vector Clustering

Despite the results we found in SVC literature about these data (Ben-Hur et al., 2001; Lee and Daniels, 2005b), our SVC implementation was able to separate the three classes without allowing BSVs; obviously, we obtain better results when we allow some BSVs. Another important difference with the results found in the SVC

Iris Plant Database - Support Vector Clustering									
Type	Setosa			Versicolour			Virginica		
	P	R	F1	P	R	F1	P	R	F1
G	100%	100%	100%	69.44%	100%	81.97%	100%	56%	71.8%
GC	100%	100%	100%	97.06%	66%	78.57%	74.24%	98%	84.48%
GCS	100%	100%	100%	90.2%	92%	91.09%	91.84%	90%	90.91%
ECS	**100**%	**100**%	**100**%	**92**%	**100**%	**95.83**%	**100**%	**92.59**%	**96.16**%

Iris Plant Database - Support Vector Clustering				
	G	GC	GCS	ECS
Accuracy	85.333%	88%	94%	**97.333%**
Macroaveraging	84.587%	87.685%	93.999%	**97.328%**

Details of Support Vector Clustering instances					
Type	Kernel	q	C	softening	# runs
G	Gaussian	0.063027	1	1	2
GC	Gaussian	0.063027	0.0666667	1	2
GCS	Gaussian	0.0891501	0.0666667	0.5	1
ECS	**Exponential**	**0.0199203**	**0.0666667**	**0.5**	1

Table 9.4: *Support Vector Clustering of the Iris Plant Database. The first table shows the Precision (P), Recall (R) and F1 for each class and for each SVC instance. The second one shows the Accuracy and the Macroaveraging. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold.*

literature is that we do not need to reduce the dimensionality of this dataset;[17] nevertheless, our results (see Table 9.4) compare better with the ones found in literature.[18]
We recall that our implementation of the SVC use the secant-like kernel width generator (see subsection 6.4.1) and thanks to it we found a suitable kernel width value for each SVC instance. Moreover, thanks to the *softening strategy* we introduced in the secant-like algorithm (see subsubsection 6.4.1.4), we found better kernel width values which led us to very good clustering results in this case.
Our *soft margin parameter* estimation discussed in section 6.5 also helped us to detect a suitable value for the C parameter. Finally, by experimenting different kernels, we found that with the Exponential kernel we have the best results on these data.

[17] Both in Ben-Hur et al. (2001) and in Lee and Daniels (2005b) the dimensionality of the dataset is reduced from 4D to 2D and 3D by means of the PCA or the Sammon's nonlinear mapping.

[18] In Ben-Hur et al. (2001) the authors obtain 14 misclassification using all four features. By using only the first three principal components, they obtain only 4 misclassifications. The latter is the same results we obtain on the complete data by means of the Exponential kernel and our softening strategy for kernel width generation.

9.5.2 Wisconsin Breast Cancer Database

In Camastra (2005) and in Lee and Daniels (2005b) we can find experimental results about these data. In the first work we find a term of comparison for the results we obtained with the Bregman Co-clustering, whereas in the second publication we find results for a comparison with the SVC.
As stated in Lee and Daniels (2005b), the Wisconsin Breast Cancer Database have 16 points with missing values. Both Camastra (2005) and Lee and Daniels (2005b) remove such points from the dataset; on the contrary we retain such points because the two clustering algorithms used here are able to deal with missing values (as we will show with the missing values experiments in the sequel).
In the following subsections we first present the results obtained with the Bregman framework both with and without feature clustering. The number of requested feature clusters is quite low with respect to the number of features. This is due to several reasons. In the first place, we want to test the effectiveness of the compression along the columns of the data matrix. In the second place, this specific case is characterized by a very sparse matrix. Generally a very high compression on such type of matrices is very useful to obtain better results. Finally, we will often use a small number of feature clusters over the whole experimental stage, for making the evaluation of clustering results easier.
The Support Vector Clustering will show the effectiveness of our contributions once again. Both the soft margin parameter estimation and the *softening strategy* have improved the results obtained with the "basic" SVC.

9.5.2.1 Bregman Co-clustering

The behavior of the Bregman framework without feature clustering (see Table 9.5) is quite different with respect to the K-means performance[19] reported in Camastra (2005). We exclude that such difference is due to the 16 points with missing values that we have retained: the percentage of missing values is too low to be influent. Rather, the two datasets could be different due to some processing of the data; moreover, the strategy for centroid initialization may be different.
We can see that the schemes $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_4$ and $\mathcal{C}_6$ perform the same with the Euclidean distance and give the best results with such Bregman divergence. A different situation is shown for the I-divergence: all the six schemes have a different performance. More precisely, the results of the schemes from 1 to 5 are different but very similar, whereas the sixth scheme outperforms all the other ones.
Also in this case, the Bregman co-clustering with the feature clustering enabled (three feature clusters) improves the overall behavior, as we can see in Table 9.6. The best approximation scheme for the Euclidean Co-clustering is the sixth scheme, whereas for the Information-Theoretic Co-clustering the best approximation scheme is the fifth one.

[19] Overall accuracy: 96.1%

W. Breast Cancer - Euclidean - No feature clustering				
Basis	Benign cancers		Malignant cancers	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_{1,2,4,6}$	**75.546**%	**32.422**%	**31.12**%	**59.893**%
$\mathcal{C}_3$	53.275%	34.933%	45.643%	66.049%
$\mathcal{C}_5$	48.69%	34.024%	52.282%	65.097%

W. Breast Cancer - Information-Theoretic - No feature clustering				
Basis	Benign cancers		Malignant cancers	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_1$	67.467%	33.833%	34.44%	64.224%
$\mathcal{C}_2$	33.188%	35.043%	65.975%	65.806%
$\mathcal{C}_3$	50.655%	36.438%	44.813%	67.665%
$\mathcal{C}_4$	32.314%	35.652%	65.975%	66.098%
$\mathcal{C}_5$	45.415%	35.202%	53.112%	66.138%
$\mathcal{C}_6$	**90.83**%	**2.118**%	**96.266**%	**15.328**%

W. Breast Cancer - Euclidean - No feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_2$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	**60.229**%	**60.229**%	27.649%	**60.229**%	33.146%	**60.229**%
W. Breast Cancer - Information-Theoretic - No feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_2$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	56.08%	44.492%	48.641%	43.92%	48.069%	**92.704**%

Table 9.5: *Bregman Co-clustering of the Wisconsin Breast Cancer Database, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the overall Accuracy. The best results are highlighted in bold.*

9.5.2.2 Support Vector Clustering

In Lee and Daniels (2005b) the Wisconsin Breast Cancer Database was employed for a case study about outliers handling. In fact, our first try was performed without allowing outliers ($C = 1$) and the data were not separated. Fortunately, our heuristics for the soft margin constant estimation has been useful also in this case. With the C value generated thanks our heuristics we obtain a better result than one obtained by Lee and Daniels (2005b). Anyway, the result in Lee and Daniels (2005b) is expressed only in terms of the recall (or completeness) for the benign class,[20] and such a value does not give much information about the overall behavior of the SVC in their case.
Our results improved when we employed our softening strategy; we obtained the best result by further hand-tuning the generated C value. See Table 9.7 for major details.

9.5.3 Wine Recognition Database

The results of both algorithms tested on the Wine Recognition Database are very similar, even though the results of the co-clustering with feature clustering en-

[20] Recall/completeness: 93.5%

W. Breast Cancer - Euclidean - With feature clustering				
Basis	Benign cancers		Malignant cancers	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_{1-5}$	75.546%	32.422%	31.12%	59.893%
$\mathcal{C}_6$	**97.817**%	**16.574**%	**63.071**%	**6.173**%
W. Breast Cancer - Information-Theoretic - With feature clustering				
Basis	Benign cancers		Malignant cancers	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_1$	66.812%	34.052%	34.44%	64.681%
$\mathcal{C}_2$	67.249%	33.906%	34.44%	64.378%
$\mathcal{C}_3$	88.21%	2.415%	95.851%	18.947%
$\mathcal{C}_4$	66.812%	33.909%	34.855%	64.407%
$\mathcal{C}_5$	**90.83**%	**2.576**%	**95.436**%	**15.441**%
$\mathcal{C}_6$	71.616%	19.012%	68.05%	44.218%

W. Breast Cancer - Euclidean - With feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_2$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	60.229%	60.229%	60.229%	60.229%	60.229%	**85,837**%
W. Breast Cancer - Information-Theoretic - With feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_2$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	55.651%	55.937%	90.844%	55.794%	**92.418**%	70.386%

Table 9.6: *Bregman Co-clustering of the Wisconsin Breast Cancer Database, with three feature clusters requested. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the overall Accuracy. The best results are highlighted in bold.*

abled are slightly better. Once again, the peculiarities of both algorithms and the effectiveness of our contributions to the SVC are confirmed.

The Bregman co-clustering yielded very good results, especially with feature clustering. On the contrary, the "basic" SVC compares worse with the Bregman co-clustering, but our contributions allowed to obtain improved performance. Both the softening strategy and the soft margin parameter estimation worked fine in these experiments, allowing us to obtain a good separation of the classes. Finally, by further hand-tuning the soft margin parameter, we obtained our best SVC results on Wine data.

9.5.3.1 Bregman Co-clustering

The first thing we noted while testing the Bregman co-clustering on the Wine Recognition Database is the behavior of the instances with the I-divergence on data with items described by negative-valued features. Theoretically, Information-Theoretic co-clustering cannot deal with negative matrices because it assumes that the input data matrix is a joint probability distribution. If we force the execution on such data, the Bregman co-clustering with the I-divergence always behaves in the same way: it equally distributes the points over all the co-cluster

W. Breast Cancer - Support Vector Clustering				
Type	Benign cancers		Malignant cancers	
	Completeness	Contamination	Completeness	Contamination
GC	98.4716%	11.0865%	79.2531%	3.66492%
GCS	97.8166%	3.44828%	93.361%	4.25532%
GCS+	**94.6058**%	**2.82**%	**97.8166**%	**4.2017**%

W. Breast Cancer - Support Vector Clustering			
	GC	GCS	GCS+
Accuracy	91.8455%	96.2804%	**96.7096%**

Details of Support Vector Clustering instances					
Type	Kernel	q	C	softening	# runs
GC	Gaussian	0.386516	0.0143062	1	2
GCS	Gaussian	0.306167	0.0143062	0.5	3
GCS+	**Gaussian**	**0.360661**	**0.0147062**	**0.5**	**3**

Table 9.7: *Support Vector Clustering of the Wisconsin Breast Cancer Database. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold.*

requested, leading to a completely meaningless partitioning.[21]
The Euclidean co-clustering worked fine also in this case, especially with feature clustering enabled. According to Table 9.9, most of the schemes yielded good results with the feature clustering, and the $\mathcal{C}_6$ scheme outperformed all other ones. Without feature clustering, only the $\mathcal{C}_6$ scheme worked fine; all other schemes yielded poor results. Such a behavior can be also found in previous works: in fact, the results of the $\mathcal{C}_2$ scheme with Euclidean distance (or, equivalently, the K-means) agree with previous results obtained with the K-means. For example, Nasser et al. (2006) obtained 49.43% of accuracy with the K-means, which is very similar to the results of the $\mathcal{C}_2$ scheme (see Table 9.8).
As already stated, no Information-Theoretic co-clustering results are available.

9.5.3.2 Support Vector Clustering

Once again, a test confirmed the effectiveness of our contributions to the SVC algorithm. The Wine data are not linearly separable, therefore we ran the SVC with a C value lower than 1, calculated thanks to our soft margin parameter estimation. This led us to acceptable results, but they were not as good as the Bregman co-clustering ones. By employing our softening strategy, the results improved significantly and by further hand-tuning the C value we obtained our best result with SVC on these data. Anyway, according to Table 9.10, the SVC best results are slightly lower than the best results obtained with the Bregman co-clustering.

[21] Due to this behavior, the results of the Information-Theoretic co-clustering will not be reported if the input dataset is negative-valued, like in this case.

Wine Recognition Database - Euclidean - No feature clustering									
Basis	First wine			Second wine			Third wine		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_1$	88.64%	66.10%	75.73%	61.64%	63.38%	62.5%	31.15%	39.58%	34.86%
$\mathcal{C}_{2,4}$	88.64%	66.10%	75.73%	36.07%	30.99%	33.33%	38.36%	58.33%	46.28%
$\mathcal{C}_3$	32%	40.68%	35.8%	31.82%	29.58%	30.66%	18.92%	14.58%	16.47%
$\mathcal{C}_5$	28.57%	23.73%	25.93%	40.58%	39.44%	40%	28.33%	35.42%	31.48%
$\mathcal{C}_6$	**90.16**%	**93.22**%	**91.67**%	**93.55**%	**81.69**%	**87.22**%	**87.27**%	**100**%	**93.20**%

Wine Recognition Database - Euclidean - No feature clustering					
	$\mathcal{C}_1$	$\mathcal{C}_{2,4}$	$\mathcal{C}_3$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	57.865%	50%	27.649%	33.146%	**90.449%**
Macroaveraging	57.697%	51.781%	29.213%	32.469%	**90.696%**

Table 9.8: *Bregman Co-clustering of the Wine Recognition Database, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. The best results are highlighted in bold.*

9.5.4 Quadruped Mammals

These data are quite simple to separate. In fact, the goal of this test is to verify the behavior of both techniques when they deal with very large dataset with a number of attributes much greater than $15/20$.[22] Both clustering methods reach optimal results separating the four classes, and outperform the CLASSIT algorithm which the data were created for (Gennari et al., 1989).
As far as the Bregman co-clustering is concerned, we have a further confirmation of its ability to deal with high dimensional data by means of the contextual feature clustering. For the SVC we have a different situation. This time no BSVs were required and the employing of the softening strategy did not make any difference in the final results.[23] On the contrary, we gained benefit by employing the Laplacian kernel.

9.5.4.1 Bregman Co-clustering

The Information-Theoretic co-clustering yielded definitely meaningless partitioning, notwithstanding the data are positive-valued. Therefore, we will not quote such results in the sequel. The point is different for the Euclidean co-clustering. It behaved quite well without feature clustering (see Table 9.11), especially the schemes $\mathcal{C}_1$, $\mathcal{C}_2$ (that equals to K-means), and $\mathcal{C}_4$. However, as already happened in other experiments, the results were significantly improved by enabling the contextual feature clustering (see Table 9.12). Again, these results prove the practical effectiveness of the theorized co-clustering ability handling high dimensional

[22] As stated in section 3.6, the effects of the *Curse of dimensionality* starts to be severe for dimensions greater than $15/20$.

[23] We have discovered that the softening strategy generally does not bring any benefit in when the data at hand are fairly easy to split.

Wine Recognition Database - Euclidean - With feature clustering									
Basis	First wine			Second wine			Third wine		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_1$	93.22%	93.22%	93.22%	92.98%	74.65%	82.81%	77.42%	100%	87.23%
$\mathcal{C}_{2,4}$	75.71%	89.83%	82.2%	89.48%	71.83%	79.69%	94.12%	100%	96.97%
$\mathcal{C}_3$	12.12%	8.47%	10%	9.84%	8.45%	9.09%	61.84%	97.92%	75.81%
$\mathcal{C}_5$	9.09%	8.47%	8.77%	20.59%	19.72%	20.14%	87.27%	100%	93.20%
$\mathcal{C}_6$	**98.28**%	**96.61**%	**97.44**%	**96.92**%	**88.73**%	**92.65**%	**87.27**%	**100**%	**93.20**%

Wine Recognition Database - Euclidean - With feature clustering					
	$\mathcal{C}_1$	$\mathcal{C}_{2,4}$	$\mathcal{C}_3$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	87.64%	85.393%	32.584%	37.64%	**94.382%**
Macroaveraging	87.768%	86.276%	31.633%	40.707%	**94.429%**

Table 9.9: *Bregman Co-clustering of the Wine Recognition Database, with three feature clusters requested. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. The best results are highlighted in bold.*

data very well.

9.5.4.2 Support Vector Clustering

The SVC with the Gaussian kernel was not able to separate the Quadruped Mammals data in the right way. More precisely, in this case each of the four clusters were split into several sub-clusters, i.e. the SVC with Gaussian kernel was not able to analyze the data at a low level of detail. Fortunately, our SVC software allows to employ a number of kernels. Among them, the Laplacian kernel led us to very good results (see Table 9.13) that are slightly better than the ones obtained by means of the scheme $\mathcal{C}_5$ of the Euclidean co-clustering.
This is the first experiment that proves the effectiveness of the Laplacian kernel on normalized or scaled data. In the sequel, we will present some other experiments where the Laplacian kernel will yield the best results on normalized/scaled data.

9.5.5 Mushroom Database

In this experiment the Bregman co-clustering does not perform as well as in the previous tests. The overall accuracy was always under the 80% for all approximation schemes, for each Bregman divergence, with or without feature clustering. On the other hand, the SVC greatly outperforms the Bregman co-clustering in this case.

9.5.5.1 Bregman Co-clustering

The Bregman co-clustering instances with the I-divergence yielded meaningless clustering results also in this experiment. For all schemes the overall accuracy was about the 50%, that shows the inability of these instances to deal with the

Wine Recognition Database - Support Vector Clustering									
Type	First wine			Second wine			Third wine		
	P	R	F1	P	R	F1	P	R	F1
GC	98,2%	91,53%	94,75%	91,4%	74,65%	82,18%	75%	100%	85,71%
GCS	94,4%	86,44%	90,25%	88,6%	87,34%	87,97%	88,89%	100%	94,12%
GCS+	**98,3**%	**96,61**%	**97,45**%	**95,4**%	**87,34**%	**91,2**%	**88,68**%	**97,92**%	**93,07**%

Wine Recognition Database - Support Vector Clustering			
	GC	GCS	GCS+
Accuracy	87.07%	90.4%	**93.26%**
Macroaveraging	87.55%	90.78%	**93.91%**

Details of Support Vector Clustering instances					
Type	Kernel	q	C	softening	# runs
GC	Gaussian	0.0613891	0.0561798	1	2
GCS	Gaussian	0.0613891	0.0561798	0.5	2
GCS+	**Gaussian**	**0.0613891**	**0.0661798**	**0.5**	**2**

Table 9.10: *Support Vector Clustering of the Wine Recognition Database. The first table shows the Precision (P), Recall (R) and F1 for each class and for each SVC instance. The second one shows the Accuracy and the Macroaveraging. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold.*

data at hand. Due to these reasons, it is useless to show the results obtained by the various Information-Theoretic co-clustering instantiations.
Once again the Euclidean co-clustering instances showed to be the more ubiquitous ones, by demonstrating their ability to deal with a large variety of data (see Table 9.14 and Table 9.15). Unfortunately, this time their performance kept low also with the feature clustering enabled: in the best case, the accuracy is less than 80% and the contamination of each class is too large.
We are sure that the problem is not the increased dimensionality of the dataset (and the subsequent experiments will confirm this) with respect to the dimensionality of the datasets used in the previous experiments. A possible explanation for this behavior is the nature of these data: the modified version of the Mushroom Database obtained from LIBSVM repository has binary-valued items and such type of data generally require *ad hoc* divergences. In fact, clustering algorithms like the K-means does not behave very well on binary-valued data, due to the inadequacy of the squared Euclidean distance (Leisch et al., 1998). This reason would explain also the poor results obtained with the I-divergence, that is a divergence studied to deal with probability-like data.

9.5.5.2 Support Vector Clustering

The SVC did not need particular tuning: no softening strategy and kernels other than the Gaussian one were needed. Instead, our soft parameter estimation helped us once again. The results achieved by SVC really differ from the ones obtained with Bregman co-clustering and reach a very good accuracy e very low contami-

Quadruped Mammals - Euclidean - No feature clustering								
Basis	Giraffe		Dog		Cat		Horse	
	Sen.	Spe.	Sen.	Spe.	Sen.	Spe.	Sen.	Spe.
$\mathcal{C}_{1,2,4}$	**70.68**%	**91.09**%	**73.18**%	**90.28**%	**92.72**%	**99.15**%	**97.51**%	**97.55**%
$\mathcal{C}_3$	21.04%	79.62%	33.47%	75.77%	22.80%	77.61%	20.06%	70.41%
$\mathcal{C}_5$	23.94%	73.77%	20.34%	70.68%	15.76%	72.15%	16.29%	75.49%
$\mathcal{C}_6$	100%	66.773%	0.08%	33.182%	0%	100%	0%	100%

Quadruped Mammals - Euclidean - No feature clustering				
	$\mathcal{C}_{1,2,4}$	$\mathcal{C}_3$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	**83.533**%	24.341%	19.065%	24.9%

Table 9.11: *Bregman Co-clustering of the Quadruped Mammals, without feature clustering. The first table shows the Sensitivity (Sen.) and the Specificity (Spe.) for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

nation for both classes.

The experiment in the previous subsection and this one show that the early conjectures about the inability of the SVC to deal with high dimensional data are probably groundless.

9.5.6 Internet advertisements

This is the last experiment of the so-called "preliminary experimental stage". The peculiarity of these data is the much greater dimensionality with respect to the other datasets used till now. The nature of these data is quite similar to the one of the previous data: the majority of the features are binary-valued, except the first three ones that assume values in a continuos range. Such continuous features make the difference in the case of the Bregman co-clustering.

Anyway, neither the Bregman co-clustering nor the SVC yielded great results for the "ads" class.

9.5.6.1 Bregman Co-clustering

Once again, only the Euclidean co-clustering instances provided good results. Since most of the 1555 features out of 1558 are binary-valued and the I-divergence is less robust with respect to data that do not assume probability-like values, the Information-Theoretic instances produced very poor results, with or without feature clustering.

The Euclidean co-clustering yielded quite good results instead. This time the best results are the same with or without feature clustering, though the schemes that yielded the best results are different in each case (see Table 9.17 and Table 9.18).

Quadruped Mammals - Euclidean - With feature clustering								
Basis	Giraffe		Dog		Cat		Horse	
	Sen.	Spe.	Sen.	Spe.	Sen.	Spe.	Sen.	Spe.
$\mathcal{C}_1$	70.68%	91.09%	73.18%	90.28%	92.72%	99.15%	97.51%	97.55%
$\mathcal{C}_{2,6}$	100%	66.77%	0.08%	33.18%	0%	100%	0%	100%
$\mathcal{C}_3$	99.92%	99.97%	99.92%	99.97%	100%	66.67%	0%	100%
$\mathcal{C}_4$	100%	66.77%	0%	100%	99.92%	99.97%	100%	99.97%
$\mathcal{C}_5$	**99.92**%	**99.97**%	**99.92**%	**99.97**%	**99.92**%	**99.97**%	**100**%	**99.97**%

Quadruped Mammals - Euclidean - With feature clustering					
	$\mathcal{C}_1$	$\mathcal{C}_{2,6}$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$
Accuracy	83.533%	24.9%	75.04%	75%	**99.92**%

Table 9.12: *Bregman Co-clustering of the Quadruped Mammals, with four features clusters. The first table shows the Sensitivity (Sen.) and the Specificity (Spe.) for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

9.5.6.2 Support Vector Clustering

We tested several SVC setups (different combinations of kernels, soft margin constraints, kernel width values) in this experiment, but no one was able to separate the non-ads elements from the ads ones. A first reason for this difficulty behavior could be the high dimensionality of the dataset; another reason could also be the nature of data.

Anyway, we took the opportunity for testing the effectiveness of employing a Bregman matrix approximation with a smaller number of features as input data for the SVC. Unfortunately, the *Bregman Co-cluster* software we usually use for testing all the schemes of the Bregman co-clustering does not output the various matrix approximations built during the clustering process, so we fell back to the *Co-cluster* software. The latter provides a verbose log of the clustering process, including each matrix approximation produced for each co-cluster update. This software provides only the schemes $\mathcal{C}_2$ and $\mathcal{C}_6$ for the Euclidean distance and we chose the latter because it yields the second best results.

Building the approximated matrix was simple: we asked the software to cluster the data with the scheme $\mathcal{C}_6$ of the Euclidean co-clustering, producing 3279 row clusters and 3 column clusters. In such a way, we were able to obtain a matrix with the same number of rows of the original matrix but with only three columns. It is important to observe that the feature compression implies a real compression of the whole data matrix, therefore the elements of the approximated matrix assume values in a smaller range and the items are not binary-valued anymore.

Once we obtained such "compressed" matrix, we fed the SVC with it and obtained results very similar to the ones obtained with Bregman co-clustering, even slightly better (see Table 9.19). This is an early evidence that the matrix compression performed by means of the Bregman co-clustering can be also used to preprocess data before input them in different clustering algorithms.

Quadruped Mammals - Support Vector Clustering								
Type	Giraffe		Dog		Cat		Horse	
	Sen.	Spe.	Sen.	Spe.	Sen.	Spe.	Sen.	Spe.
LC	**100**%	**100**%	**100**%	**99.9734**%	**99.9208**%	**99.9465**%	**99.8395**%	**100**%

Quadruped Mammals - Support Vector Clustering	
	LC
Accuracy	**99.94**%

Details of Support Vector Clustering instances					
Type	Kernel	q	C	softening	# runs
LC	**Laplacian**	**0.0395438**	**1**	**1**	**1**

Table 9.13: *Support Vector Clustering of the Quadruped Mammals. The first table shows the Sensitivity (Sen.) and the Specificity (Spe.) for each class and for each co-clustering basis. The second one shows the Accuracy. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold.*

9.5.7 Conclusion

A preliminary experimental stage was performed over a number of datasets, different for number of items, number of features, data type, etc. Therefore, we can draw various conclusions. Above all, we verified the ability of the algorithms to deal with a number of application domains. This flexibility was already theorized in the previous chapters: the Bregman co-clustering can employ different statistical approximation schemes and various divergences to achieve that goal, whereas each iteration of the SVC analyzes the data at different levels of detail and can discover the most suitable latent data structure.
Next, the Bregman co-clustering showed its ability of improving general performance thanks to the contextual feature clustering. The same reason also explains the ability of the Bregman co-clustering to deal with high dimensional data. The latter was also proved in this test phase.
Finally, even though the I-divergence outperformed the Euclidean distance in some experiments, the Euclidean divergence resulted more robust on average.
As far as the SVC is concerned, we showed the effectiveness of our contributions to this clustering technique. Moreover, we also proved the ability of this clustering algorithm to deal with overlapping clusters and high dimensional data. Finally, we proved the effectiveness of employing different kernels in some particular experiments.
Till now both algorithms have showed very good performance, most of the time quite similar. The SVC provides the best results tested on *Iris Plant Database*, *Wisconsin Breast Cancer Database*, *Quadruped Mammals*,[24] *Mushroom Database*. On the other hand, the Bregman co-clustering provides the best results tested on *Wine Recognition Database* and *Internet advertisements*.

[24]The practical difference between the SVC results and Bregman Co-clustering results does not exist. See Table 9.20.

Mushroom Database - Euclidean - No feature clustering				
Basis	Edible		non-Edible	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_1$	48.979%	52.619%	49.382%	49.019%
$\mathcal{C}_2$	50.511%	51.088%	50.903%	47.5%
$\mathcal{C}_3$	49.081%	52.26%	50%	48.658%
$\mathcal{C}_4$	49.438%	52.491%	49.168%	48.901%
$\mathcal{C}_5$	50.715%	51.24%	50.404%	47.643%
$\mathcal{C}_6$	**87.513**%	**31.157**%	**63.142**%	**15.544**%

Mushroom Database - Euclidean - No feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	49.188%	50.714%	49.557%	49.298%	50.554%	**74.889**%

Table 9.14: *Bregman Co-clustering of the Mushroom Database, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

According to Table 9.20, the test where the SVC totally outperforms the Bregman co-clustering is the *Mushroom database*. On the contrary, the test where Bregman co-clustering is definitely better than the SVC is the Internet advertisements, alothough the SVC yielded slightly better results using the Bregman approximated matrix.
The next experimental session aims at verifying the ability of the SVC to deal with datasets containing a great number of outliers. At the same time we will also verify the inability of the Bregman co-clustering to deal with the same kind of data.

9.6 Outliers handling: the data

This experimental stage is intended for proving the ability of the SVC to deal with outliers. The theory and some experiments provided by other researchers state that the SVC can discover the most suitable latent structure by itself without suffering the influence of the outlier points. This should be a distinctive characteristic of the SVC with respect to the Bregman co-clustering.[25] In fact, a limit of the alternate minimization scheme is the inability to neglect outlier points. Moreover such a clustering scheme negatively suffers the influence of such kind of points.
The two datasets we present in the following (see Table 9.21) are synthetic datasets built by means of *SynDECA*, a tool to generate synthetic datasets for evaluation of clustering algorithms (Vennam and Vadapalli, 2005).
The clusters of both datasets are simple to separate without outliers in order to be sure that the trend of the results is only influenced by the outliers.

[25] Actually, as already stated in subsection 5.10.1, a very new extension of the Bregman framework (the Bregman Bubble Co-clustering) should be able to deal with outliers. Anyway, this new extended framework was made public while we were finalizing the work on this thesis, therefore we were not able to test it.

Mushroom Database - Euclidean - With feature clustering				
Basis	Edible		non-Edible	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_1$	50.485%	51.245%	50.618%	47.653%
$\mathcal{C}_{2,3}$	86.261%	31.868%	62.452%	16.993%
$\mathcal{C}_4$	**88.202**%	**31.085**%	**62.975**%	**14.846**%
$\mathcal{C}_5$	87.64%	31.195%	63.023%	15.434%
$\mathcal{C}_6$	76.609%	45.848%	39.639%	35.449%

Mushroom Database - Euclidean - With feature clustering					
	$\mathcal{C}_1$	$\mathcal{C}_{2,3}$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	50.554%	73.929%	**75.135**%	74.889%	57.459%

Table 9.15: *Bregman Co-clustering of the Mushroom Database, with three feature clusters requested. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

9.6.1 Syndeca 02

The first dataset used for this stage is called *Syndeca 02* and has 1000 points represented by 10 features. There are five random shaped clusters in this dataset: *A* (327 points), *B* (134 points), *C* (162 points), *D* (132 points), and *E* (133 points). The remaining 112 points are outliers and do not belong to any of the above clusters (see Figure 9.1).

9.6.2 Syndeca 03

The second dataset used for this stage is called *Syndeca 03* and has 10000 points represented by 3 features. There are six random shaped clusters in this dataset: *A* (1029 points), *B* (1688 points), *C* (1987 points), *D* (1241 points), *E* (1207 points), and *F* (1578 points). The remaining 1270 points are outliers and do not belong to any of the above clusters (see Figure 9.2).

9.7 Outliers handling: the results

In this section we will present the results obtained both with the SVC and with Bregman co-clustering. We will see the SVC dealing very well with data full of outliers. The Bregman co-clustering results will confirm that the framework inherits the problems of the classical K-means-like clustering scheme.
For the sake of simplicity, in the following we report just the accuracy and the macroaveraging values, since the results for each cluster do not give more useful informations due to the trivial nature of the problems. The accuracy value is calculated over all points in a dataset, so it can never be 100% because the outliers are not included in any cluster. On the contrary, the macroaveraging value is calculated as the arithmetic mean of the F1 values, therefore it can reach the 100%.

Mushroom Database - Support Vector Clustering				
Type	Edible		non-Edible	
	Completeness	Contamination	Completeness	Contamination
GC	**97.8**%	**6.22**%	**93.96**%	**2.1**%

Mushroom Database - Support Vector Clustering	
	GC
Accuracy	**95.81**%

Details of Support Vector Clustering instances					
Type	Kernel	q	C	softening	# runs
GC	**Gaussian**	**0.0277778**	**0.00123092073**	**1**	**1**

Table 9.16: *Support Vector Clustering of the Mushroom Database. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold.*

9.7.1 Support Vector Clustering

The SVC behaved very well on both Syndeca datasets, perfectly detecting each cluster. Therefore we will focus on the advantages of our contributions in these cases.
The experiments proved the expected characteristic of the SVC: the ability of creating a singleton cluster for each outlier or, anyway, a great number of clusters with a very low cardinality including few outliers. In fact, the SVC produced 110 clusters for 112 outliers in the first experiment, whereas it output 501 clusters for 1270 outliers in the second one.
As Table 9.22 shows, these experiments did not need a particular value for the soft margin parameter to achieve the best results. Theoretically, they did not need the softening strategy neither. Anyway, the softening strategy allowed us to obtain the same results with the 50% of iterations less, i.e. the softening strategy allowed the SVC to be about 50% faster in these experiments.

9.7.2 Bregman Co-clustering

Bregman co-clustering behaved as expected, suffering the presence of the outliers very much, especially running on the second dataset. In fact, with Syndeca 02 data, the cluster assignment of non-outlier points was perfectly performed by the scheme $\mathcal{C}_6$ of the Information-Theoretic co-clustering. Obviously, we calculated the accuracy only over the non-outlier points, but the clusters are contaminated by outliers because the co-clustering algorithm can only distribute such points among the requested clusters. These results (see Table 9.22) show that in this case the outliers did not influence the scheme $\mathcal{C}_6$ performance, but the clusters cannot be pure.
On the other hand, testing the Bregman co-clustering on the Syndeca 03 data

Internet advertisement - Euclidean - No feature clustering						
Basis	Ads			non-Ads		
	P	R	F1	P	R	F1
$\mathcal{C}_{1,2,4}$	36.57%	64.05%	46.56%	93.33%	81.91%	87.24%
$\mathcal{C}_3$	16.53%	42.48%	23.8%	87.42%	65.07%	74.6%
$\mathcal{C}_5$	13.12%	47.93%	20.6%	85.08%	48.33%	61.64%
$\mathcal{C}_6$	**69.47**%	**48.58**%	**57.18**%	**92.02**%	**96.52**%	**94.22**%

Internet advertisement - Euclidean - No feature clustering				
	$\mathcal{C}_{1,2,4}$	$\mathcal{C}_3$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	79.414%	61.909%	48.277%	**89.814**%

Table 9.17: *Bregman Co-clustering of the Internet advertisements, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

Internet advertisement - Euclidean - With feature clustering						
Basis	Ads			non-Ads		
	P	R	F1	P	R	F1
$\mathcal{C}_1$	36.57%	64.05%	46.56%	93.33%	81.91%	87.24%
$\mathcal{C}_{2,3}$	35.41%	64.49%	45.72%	93.33%	80.85%	86.64%
$\mathcal{C}_{4,5}$	**69.47**%	**48.58**%	**57.18**%	**92.02**%	**96.52**%	**94.22**%
$\mathcal{C}_6$	42.83%	50.76%	46.46%	91.74%	88.97%	90.34%

Internet advertisement - Euclidean - With feature clustering				
	$\mathcal{C}_1$	$\mathcal{C}_{2,3}$	$\mathcal{C}_{4,5}$	$\mathcal{C}_6$
Accuracy	79.414%	78.561%	**89.814**%	83.623%

Table 9.18: *Bregman Co-clustering of the Internet advertisements, with three feature clusters requested. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

shown how a great number of outliers can trouble the clustering process and drastically reduce its performance.
We also tried to ask for more clusters in order to test if the outliers were isolated in different clusters. Unfortunately, the outliers influence the clustering process anyway, so the results did not significantly change. On the contrary, in some cases the results got worse, due to the increased cardinality of the requested partitioning that induce the algorithm to distribute all points over a greater number of clusters.

9.7.3 Conclusion

The tests on the outliers handling yielded expected results, especially for the Bregman Co-clustering. As already said many times, the inability of dealing with this kind of data is an intrinsic limitation of the classical alternate minimization scheme adopted by Bregman Co-clustering as well as by the classical K-means al-

Internet advertisement (compressed) - Support Vector Clustering						
Type	Ads			non-Ads		
	P	R	F1	P	R	F1
GC	**47,28**%	**78,90**%	**59,13**%	**97,94**%	**91,94**%	**94,85**%

Internet advertisement (compressed) - Support Vector Clustering	
	GC
Accuracy	**90,85**%

Details of Support Vector Clustering instances					
Type	Kernel	q	C	softening	# runs
GC	**Gaussian**	**616.886**	**0.00104971**	**1**	**9**

Table 9.19: *Support Vector Clustering of the Internet advertisements. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold.*

Dataset	SVC	Bregman Co-clustering
Iris Plant Database	97.333%	96.667%
W. Breast Cancer Database	96.71%	92.418%
Wine Recognition Database	93.26%	94.382%
Quadruped Mammals	99.94%	99.92%
Mushroom Database	95.81%	75.135%
Internet advertisement	-	89.814%
Internet advertisement (compressed)	90.85%	-

Table 9.20: *Summary of the preliminary stage results. Only the (best) accuracy reported.*

gorithm. If the number of the outliers is not so large, the Bregman Co-clustering preserve its ability of putting most of the objects in their right cluster, but it cannot avoid to contaminate clusters with outlier points that are supposed to not belong anywhere. This limitation should be overcame with the cutting edge of the Bregman framework, i.e. the Bregman Bubble Co-clustering (Deodhar et al., 2007a). Unfortunately, we were not able to test this new generalization of the Bregman framework, due to two main reasons: (i) it was completed and published only while we were finalizing our work on this thesis, and (ii) there is no software publicly available.

The behavior of the SVC was predicted by the theory and has been successfully proved with this experimental session. The ability of assigning the points to the most suitable clusters was not affected by the presence of the outlier points; moreover, such points were isolated in low-cardinality clusters or singleton ones.

Dataset	# of items (n)	# dimensions (d)	# of classes	# of outliers
Syndeca 02	1000	10	5	112
Syndeca 03	10000	3	6	1270

Table 9.21: *Summary of datasets used in the outlier handling stage of experiments.*

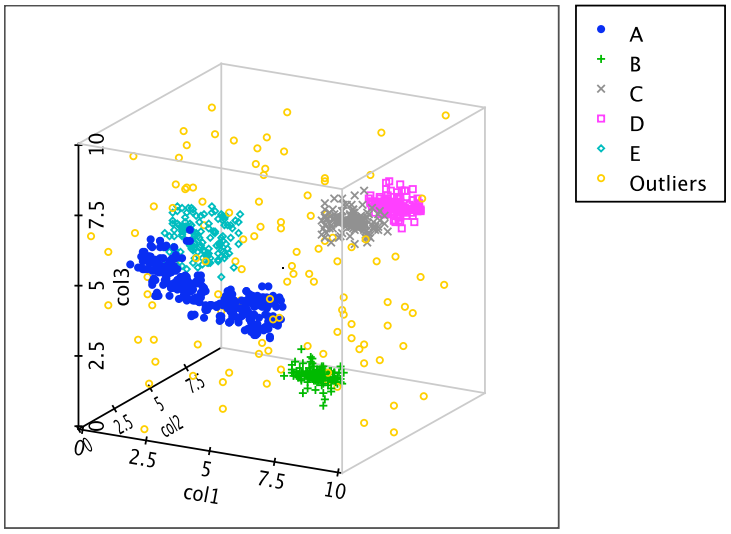


Figure 9.1: *The* Syndeca 02 *3D plot was made by means of the first three attributes. Since we asked the SynDECA generator to produce a fairly separable problem, we were sure that the plot would be faithful with any feature. Since the features are unnamed, in the plot we wrote "col1", "col2", and "col3" for the first, the second, and the third feature respectively. The outliers are represented by empty circles.*

9.8 Missing Values in Astrophysics: the data

The missing values problem was faced within the astrophysics application domain, using star/galaxy datasets. The source of our data is the Sloan Digital Sky Survey (SDSS), the most ambitious astronomical survey ever undertaken.[26] When completed, it will provide detailed optical images covering more than a quarter of the sky, and a 3-dimensional map of about a million galaxies and quasars. As the survey progresses, the data are released to the scientific community and the general public in annual increments. More specifically, the data we used in the experiments are from the sixth Data Release (DR). The photometric data in DR6 are based on five-band imaging observations of 8520 square degrees of sky, and include measures of 300 million unique objects.
The missing values experiments were divided in two different sessions. In the first one, the data were extracted with specific features having missing values, following the same SDSS recommendations used by Wagstaff (2004) and Wagstaff and Laidler (2005), in order to obtain a "clean" sample and exclude faint stars (flux of > 20 magnitude) and very bright galaxies (flux of <= 20 magnitude). Each

[26] See `http://www.sdss.org` for major details.

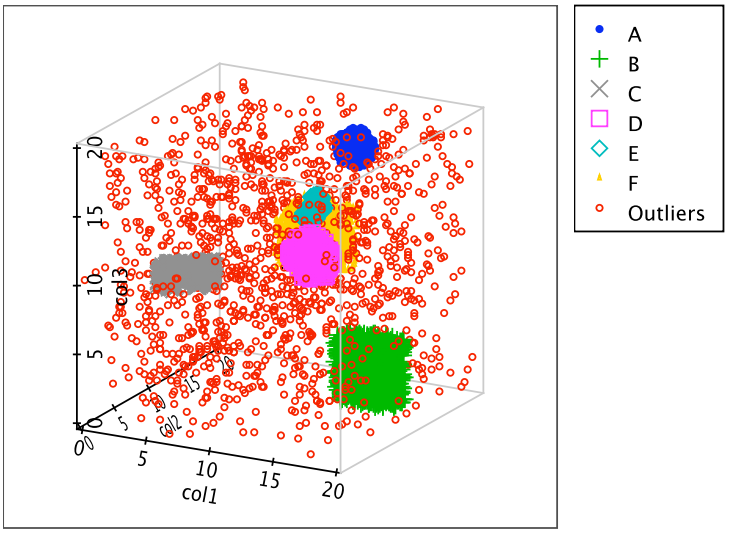


Figure 9.2: *Since the dataset is 3-dimensional, this plot exactly represents* Syndeca 03 *data. The features are unnamed, so in the plot we have written "col1", "col2", and "col3" for the first, the second, and the third feature respectively. The outliers are represented by empty circles.*

object is represented by 25 features:[27] brightness (Point Spread Function flux), size (Petrosian radius, in arcsec), texture (small-scale roughness of object), and two shape features, all observed at 365 nm. The two shape features (10 actual attributes) are the ones that can have missing values.
The guidelines used to extract the objects from the SDSS allowed us to get data that are fairly separable based on the features we used, so we were able to focus on and accurately test the missing values robustness of our two clustering algorithms.
In the second session data were extracted in a similar way, but the actual features are 15 because we avoided features with missing values; then we created different missing-valued versions of the various datasets by means of a simple homemade program that removes values on specific features. The items affected by this procedure are randomly chosen and their number is specified as an input parameter.
Finally, to evaluate our clustering results we considered an object to be a Star if the "type_u, type_g, type_r, type_i, type_z" attributes in the SDSS catalogue assume the value 6. An object is a Galaxy if the aforesaid attributes assume the value 3.[28]

[27] Five attributes, each over the u, g, r, i, z bands.

[28] The "type" attributes assume values in function of the Stellarity index returned by the SExtractor. SExtractor (Source-Extractor) is a program that builds a catalogue of objects from an astronomical image. It is particularly oriented towards reduction of large scale galaxy-survey data, but it also performs well on moderately crowded star fields. For more information see `http://sextractor.sourceforge.net/`.

Outliers experiments: Syndeca 02 e Syndeca 03 data								
	Syndeca 02				Syndeca 03			
	GC	GCS	EuCC $\mathcal{C}_1$	ITCC $\mathcal{C}_6$	GC	GCS	EuCC $\mathcal{C}_1$	ITCC $\mathcal{C}_6$
Accuracy	88.8%	88.8%	68.4%	88.8%	87.3%	87.3%	39.47%	47.7%
Macroavg	100%	100%	63.48%	94.18%	100%	100%	39.90%	49.00%

Details of Support Vector Clustering instances									
		Syndeca 02				Syndeca 03			
Type	Kernel	q	C	softening	# runs	q	C	softening	# runs
GC	Gaussian	0.599011	1	1	8	4.49681	1	1	16
GCS	Gaussian	0.506759	1	0.5	4	2.5919	1	0.5	7

Table 9.22: *Support Vector Clustering and Bregman Co-clustering of* Syndeca 02 *and* Syndeca 03 *data. The first table shows the Accuracy and the Macroaveraging both for SVC and for Bregman Co-clustering. For the latter, only the best schemes are reported ($\mathcal{C}_1$ for the Euclidean co-clustering and $\mathcal{C}_6$ for the Information-Theoretic co-clustering). The second table is a legend that provides details about the SVC instances.*

Dataset	# of items (n)	# of features affected	% of missing-valued items
MVSG5000	5000	10	27%
MVSG10000	10000	10	29%

Table 9.23: *Summary of datasets used in the first session of the missing values clustering experiments. Both datasets are 25-dimensional and the actual attributes reporting missing values are the last 10. The data are divided in two classes: Star and Galaxy.*

9.8.1 Data details

We refer to data of the first session as *Missing-Valued Stars and Galaxies (MVSG)* data, whereas the data in the second session will be indicated as *Artificial Missing-Valued Stars and Galaxies (AMVSG)*. Therefore, we have two datasets per session: MVSG5000 and MVSG10000 in the first session; AMVSG5000 and AMVSG10000 in the second one. Each dataset was built by means of astronomical objects taken from a different slice of sky, and the number of stars and galaxies was manually rounded. The MVSG5000 dataset has 2000 stars and 3000 galaxies, whereas the MVSG10000 dataset has 4560 stars and 5440 galaxies. The AMVSG5000 has 2150 stars and 2850 galaxies, whereas AMVSG10000 has 4400 stars and 5600 galaxies. Finally, we artificially created some missing-valued versions of the AMVSG data:

- I5F3: 5% of items have missing values on three features;
- I10F3: 10% of items have missing values on three features;
- I20F3: 20% of items have missing values on three features;
- I30F3: 30% of items have missing values on three features;
- I5F6: 5% of items have missing values on six features;
- I10F6: 10% of items have missing values on six features;

Dataset	# of items (n)	# of features affected	% of missing-valued items
AMVSG5000	5000	0	0%
AMVSG10000	10000	0	0%
AMVSG5000I5F3	5000	3	5%
AMVSG10000I5F3	10000	3	5%
AMVSG5000I10F3	5000	3	10%
AMVSG10000I10F3	10000	3	10%
AMVSG5000I20F3	5000	3	20%
AMVSG10000I20F3	10000	3	20%
AMVSG5000I30F3	5000	3	30%
AMVSG10000I30F3	10000	3	30%
AMVSG5000I5F6	5000	6	5%
AMVSG10000I5F6	10000	6	5%
AMVSG5000I10F6	5000	6	10%
AMVSG10000I10F6	10000	6	10%
AMVSG5000I20F6	5000	6	20%
AMVSG10000I20F6	10000	6	20%
AMVSG5000I30F6	5000	6	30%
AMVSG10000I30F6	10000	6	30%

Table 9.24: *Summary of datasets used in the second session of the missing values clustering experiments. All datasets are 15-dimensional and the data are divided in two classes: Star and Galaxy.*

- I20F6: 20% of items have missing values on six features;
- I30F6: 30% of items have missing values on six features;

The above eight variants were created for both the AMVSG5000 and AMVSG10000 (see Table 9.24). For variants with three features affected we have chosen the brightness, the size and the texture only over the 'r' band. For variants with six features, the same attributes are also affected over the 'g' band. Let us remark that the original AMVSG data have 15 actual features: removing values over six features degrades the information very much. In real situations the number of missing values is usually much less than the number of missing values introduced in these data, and so the number of items affected. Moreover, the features affected in the second session have a greater informative burden than the ones reporting missing values in MVSG data.

9.9 Missing Values in Astrophysics: the results

The experiments for verifying the robustness with respect to the missing values yielded very good results. The most surprising results are the ones obtained by means of the SVC algorithm. In fact, there are no previous works or experiments about this capability of the SVC. We hypothesized it because of a recent work about SVM classifiers: Ancona et al. (2004) showed that the SVM classifiers increase their generalization capability when trained with moderated sparse data. Our experiments on missing-valued data showed that also the One Class SVM

MVSG5000 - Euclidean - No feature clustering				
Basis	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_{1,2,4,6}$	**100**%	**41.962**%	**51.8**%	**0**%
$\mathcal{C}_3$	56.25%	36.209%	51.867%	35.993%
$\mathcal{C}_5$	44.45%	33.64%	48.133%	23.483%

MVSG5000 - Euclidean - No feature clustering			
	$\mathcal{C}_{1,2,4,6}$	$\mathcal{C}_3$	$\mathcal{C}_5$
Accuracy	**71.08**%	53.62%	46.66%

Table 9.25: *Bregman Co-clustering of the MVSG5000 dataset, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

(and so the SVC) can handle sparse data very well. Therefore we can handle missing-valued data as sparse data.
On the other hand, the results of the Bregman Co-clustering were expected. The Bregman Co-clustering can cluster the missing-valued dataset thanks to two of its own characteristics. The first one is the contextual feature compression and the related matrix approximation ability: clustering is performed on approximated matrices with no missing values anymore.[29] The other peculiarity of the Bregman Co-clustering is the chance of assigning to elements a probability distribution other than the uniform one (see subsection 5.4.1). In fact, the Bregman Co-clustering formalism is so abstract that it also provides the way for specifying a custom probability distribution. Although the probability distribution is usually assumed to be uniform, this peculiarity allows us to deal with a wider variety of situations, including the modeling of matrices with missing values. Unfortunately, this possibility is not fully available in the softwares we used for the experimental stage. Therefore, we faced the missing values problem mainly by means of the contextual feature clustering, like a simple data sparseness problem.[30]
Since we can reproduce the K-means algorithm by means of the Bregman framework, we will show the advantages of the co-clustering and the SVC over such a classical clustering algorithm.
Finally, since the Star/Galaxy data contain negative values, the co-clustering instances with I-divergence was not able to work, therefore we show only the Euclidean instances results.

9.9.1 Results on MVSG data

As in the previous experiments, we provide both Bregman co-clustering (Euclidean only in this case) results and SVC ones. According to Table 9.25 and Table 9.27, the Euclidean one-way clustering (without feature clustering) did not yield good results. The schemes $\mathcal{C}_1$, $\mathcal{C}_2$ (i.e. the K-means), $\mathcal{C}_4$, and $\mathcal{C}_6$ are the ones

[29] Actually, the sparseness is kept with some divergences, such as the I-divergence.
[30] A problem involving data represented by sparse matrices.

MVSG5000 - Euclidean - With feature clustering				
Basis	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_1$	100%	41.962%	51.8%	0%
$\mathcal{C}_2$	100%	13.83%	89.3%	0%
$\mathcal{C}_3$	100%	17.999%	85.367%	0%
$\mathcal{C}_4$	**100**%	**12.968**%	**90.067**%	**0**%
$\mathcal{C}_5$	100%	16.632%	86.7%	0%
$\mathcal{C}_6$	97.65%	35.757%	63.767%	2.398%

MVSG5000 - Euclidean - With feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	71.08%	93.58%	91.22%	**94.04**%	92.02%	77.32%

Table 9.26: *Bregman Co-clustering of the MVSG5000 dataset, with three feature clusters. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

MVSG10000 - Euclidean - No feature clustering				
Basis	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_{1,2,4,6}$	**100**%	**54.561**%	**54.265**%	**0**%
$\mathcal{C}_3$	56.25%	31.92%	51.379%	35.993%
$\mathcal{C}_5$	50.088%	25.219%	47.426%	23.483%

MVSG10000 - Euclidean - No feature clustering			
	$\mathcal{C}_{1,2,4,6}$	$\mathcal{C}_3$	$\mathcal{C}_5$
Accuracy	**75.12**%	48.%	48.64%

Table 9.27: *Bregman Co-clustering of the MVSG10000 dataset, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

that performed the best, but the results are very poor anyway. The results in Table 9.26 and Table 9.28 show that the performance were improved when we enabled the feature clustering.

The results obtained by means of Bregman Co-clustering with feature clustering enabled are very good. However, the SVC surprisingly outperformed the best Euclidean co-clustering scheme when tested on the MVSG5000 dataset. SVC reached a very low contamination for the *Star* cluster and, consequently, a very high value for the overall accuracy (see Table 9.29). As far as the MVSG10000 data are concerned, the SVC results are much more similar to the best ones obtained by the Euclidean co-clustering (see Table 9.30).

As already happened in the previous tests, one of our contributions to the SVC resulted very useful also in this case; in fact, the heuristics for the soft margin parameter estimation yielded very useful values.

MVSG10000 - Euclidean - With feature clustering				
Basis	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_1$	100%	54.561%	54.265%	0%
$\mathcal{C}_2$	100%	7.675%	93.566%	0%
$\mathcal{C}_3$	100%	14.232%	88.07%	0%
$\mathcal{C}_4$	**100**%	**7.632**%	**93.603**%	**0**%
$\mathcal{C}_5$	100%	11.075%	90.717%	0%
$\mathcal{C}_6$	98.991%	41.205%	65.809%	1.285%

MVSG10000 - Euclidean - With feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	71.08%	96.5%	93.51%	**96.52**%	94.95%	80.94%

Table 9.28: *Bregman Co-clustering of the MVSG10000 dataset, with three feature clusters. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

9.9.2 Results on AMVSG data

The second session of the clustering experiments with missing-valued data is intended to verify the robustness of the algorithms with respect to a great lack of information. The "holes" in data was injected thanks to a homemade tool that randomly chooses the items to affect. The percentage of objects is user-defined, and so are the attributes. As said in the previous sections, we chose to affect those features that provide a lot of useful information for the clustering process. Both algorithms performed very well, with very similar results. The SVC performed better or worse than the Euclidean Co-clustering,[31] depending on the dataset, the percentage of objects, and the number of features affected.
The number of datasets used for this experimental session is very large,[32] therefore we choose not to report all the co-clustering bases results for the Euclidean Co-clustering, and so we also did for the SVC instances. For each dataset we report only the results of the best co-clustering basis and the best SVC instance.
According to Table 9.31 and Table 9.33, the best basis for the Euclidean one-way clustering was the $\mathcal{C}_2$ in every case. Therefore, in those tables we find K-means results. Let us remark that the Bregman clustering works quite well also without feature clustering enabled, and it is much probable that this depends on the possibility of treating the '0' values as missing values.
In Table 9.32 and Table 9.34 we can see the results of the Bregman Co-clustering with the feature compression enabled. The behavior of the Bregman co-clustering is what we expected: in most cases, the quality of the partitioning decreases as the lack information increases.
According to Table 9.35 and Table 9.36, the SVC results are not proportional to the increment of the lack of information. This is due to some reasons: above all,

[31] Also in this case the Information-Theoretic Co-clustering was not used because the data are negative-valued.
[32] Eighteen datasets (2 dataset and 8 variants for each of them).

MVSG5000 - Support Vector Clustering				
Type	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
GC	**100**%	**2.39**%	**98.3667**%	**0**%

MVSG5000 - Support Vector Clustering	
	GC
Accuracy	**99.02**%

Details of Support Vector Clustering instances					
Type	Kernel	q	C	softening	# runs
GC	**Gaussian**	**0.492154**	**0.002**	**1**	**3**

Table 9.29: *Support Vector Clustering of the MVSG5000 data. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold.*

the hyper-parameters tuning is different case by case. As far as the kernel width parameter is concerned, the secant-like algorithm have no problem to discover the most suitable value for each missing-valued variant of the data. The point is different for the soft margin parameter estimation: the increasing of the missing values in the data matrix can make our estimation heuristics less effective. The hand-tuning of such parameter was necessary in many cases, as we can see especially in Table 9.36.
In the second place, the distribution of the missing values can affect the One Class SVM performance more than the number of such missing values, therefore in some cases the quality of the clustering is better even though the number of missing values is greater.

9.9.3 Conclusion

The experimental stage to verify the robustness of the studied algorithms with respect to the missing values was divided into two sessions. The first one was about datasets with "natural" missing values, i.e. a subset of the features chosen for representing the astronomical objects was likely to have missing values. This type of test was borrowed from previous works carried out at the *California Institute of Technology (CALTECH)* (Wagstaff, 2004; Wagstaff and Laidler, 2005). The results of this first session were very satisfying.
The second session was setup to verify the aforementioned robustness by performing the tests on datasets that were artificially deprived of an increasing quantity of information. The datasets have initially no missing values, then the number of missing values was increased and several missing-valued variants of the initial data were created. The feature we chose to affect carry a lot of information, therefore a high level of missing values drastically reduced the quality of the partitioning obtained with both the Bregman co-clustering and the SVC. However,

MVSG10000 - Support Vector Clustering				
Type	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
GC	**100**%	**7,82**%	**92.886**%	**0**%

MVSG10000 - Support Vector Clustering	
	GC
Accuracy	**96.13**%

Details of Support Vector Clustering instances					
Type	Kernel	q	C	softening	# runs
GC	**Gaussian**	**0.17332**	**0.001**	**1**	**2**

Table 9.30: *Support Vector Clustering of the MVSG10000 data. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy. The third table is a legend that provides details about the SVC instances. The best results are highlighted in bold.*

the performance of the algorithms was always good related to the critical conditions.

In the end, the missing values robustness of both algorithms within the astrophysics application domain was successfully proved.

We recall that the missing values problem was faced here as a data sparseness problem. In fact, the missing values were replaced by "zero" values. This was mandatory for testing the SVC that does not explicitly support the missing values handling. On the other hand, the Bregman co-clustering software support the "missing values", but it practically fills those empty cell with zero values before starting the clustering process.

Additional proofs about the (real) data sparseness robustness of the Bregman clustering framework and the SVC will be provided in section 9.12, where we deal with text clustering problems. In fact, in addition to the high dimensionality, the sparseness is another characteristic of text documents represented as term-document matrices. A lot of words are usually used as features, but only a subset of them is present in a particular document, therefore the data matrix is often very sparse.

Dataset	**# of items** (n)	**# dimensions** (d)	**# of Stars**	**# of Galaxies**
Longo 01	9818	15	2935	6883
Longo 02	10942	15	2978	7964
Longo 03	2500	15	2000	500

Table 9.37: *Summary of datasets used in the Star/Galaxy unsupervised separation to verify the behavior with respect to strongly overlapping clusters.*

AMVSG5000 - Euclidean - No feature clustering						
Dataset	Star		Galaxy			
	Comp.	Cont.	Comp.	Cont.	Accuracy	Basis
AMVSG5000	100%	41.96%	51.8%	0%	72%	$\mathcal{C}_2$
AMVSG5000I5F3	100%	40.6%	54.43%	0%	72.7%	$\mathcal{C}_2$
AMVSG5000I10F3	99.95%	24.1%	78.83%	0.04%	87.3%	$\mathcal{C}_2$
AMVSG5000I20F3	99.95%	24.8%	78%	0.04%	86.8%	$\mathcal{C}_2$
AMVSG5000I30F3	99.95%	26.8%	75.63%	0.04%	85.4%	$\mathcal{C}_2$
AMVSG5000I5F6	99.95%	23.67%	79.33%	0.04%	87.6%	$\mathcal{C}_2$
AMVSG5000I10F6	100%	22.48%	80.67%	0%	88.4%	$\mathcal{C}_2$
AMVSG5000I20F6	100%	27.48%	74.73%	0%	84.84%	$\mathcal{C}_2$
AMVSG5000I30F6	99.95%	33.9%	65.8%	0.05%	79.46%	$\mathcal{C}_2$

Table 9.31: *Bregman Co-clustering of the AMVSG5000 dataset and related missing-valued variants, without feature clustering. The table shows the results obtained by the best co-clustering basis: the Completeness (Comp.), the Contamination (Cont.), and the Accuracy for each dataset.*

AMVSG5000 - Euclidean - With feature clustering						
Dataset	Star		Galaxy			
	Comp.	Cont.	Comp.	Cont.	Accuracy	Basis
AMVSG5000	100%	12.97%	90.07%	0%	94%	$\mathcal{C}_4$
AMVSG5000I5F3	100%	16.87%	86.47%	0%	91.88%	$\mathcal{C}_4$
AMVSG5000I10F3	100%	17.76%	85.6%	0%	91.36%	$\mathcal{C}_4$
AMVSG5000I20F3	100%	18.6%	84.73%	0%	90.84%	$\mathcal{C}_2$
AMVSG5000I30F3	100%	20.8%	82.467%	0%	89.48%	$\mathcal{C}_2$
AMVSG5000I5F6	97.1%	17.78%	86%	0.2%	90.44%	$\mathcal{C}_2$
AMVSG5000I10F6	100%	22.1%	80.7%	0%	88.6%	$\mathcal{C}_1$
AMVSG5000I20F6	100%	22.2%	80.94%	0%	88.56%	$\mathcal{C}_4$
AMVSG5000I30F6	99.95%	33.9%	65.8%	0.05%	79.46%	$\mathcal{C}_1$

Table 9.32: *Bregman Co-clustering of the AMVSG5000 dataset and related missing-valued variants, with three feature clusters. The table shows the results obtained by the best co-clustering basis: the Completeness (Comp.), the Contamination (Cont.), and the Accuracy for each dataset.*

9.10 Overlapping clusters in Astrophysics: the data

For this stage of the experiments we faced the Star/Galaxy separation problem once again.[33] The data were extracted from the same Data Release of the SDSS used in previous experimental stage and the same AMVSG data features were selected. On the contrary, different criteria were used in order to have a more real problem to tackle. We built three datasets by extracting the objects from different sky slices which are one squared degree large each (see Table 9.47). The patterns were extracted from the catalogue of the primary photometric objects by selecting

[33]More precisely, this problem is more than a star/galaxy separation. It can be traced to the separation of sources into spatially unresolved ones (following the astronomical tradition, we shall call them "stars", but physically also including quasars or possibly other types of objects whose intrinsic angular size is much smaller than the effective angular resolution of the survey), and spatially resolved ones ("galaxies", but possibly other types of extended sources). For more information about the Star/Galaxy separation problem, look it in Donalek (2006, chap. 5).

AMVSG10000 - Euclidean - No feature clustering						
Dataset	Star		Galaxy			
	Comp.	Cont.	Comp.	Cont.	Accuracy	Basis
AMVSG10000	100%	11.301%	89.32%	0%	94.19%	$\mathcal{C}_2$
AMVSG10000I5F3	100%	20%	79.04%	0%	88.6%	$\mathcal{C}_2$
AMVSG10000I10F3	99.06%	19.62%	79.724%	0.98%	88.54%	$\mathcal{C}_2$
AMVSG10000I20F3	79.96%	39.85%	55.58%	23.21%	66.7%	$\mathcal{C}_2$
AMVSG10000I30F3	70%	38.05%	63.95%	28.22%	66.71%	$\mathcal{C}_2$
AMVSG10000I5F6	93.16%	24.12%	75.18%	7.09%	83.4%	$\mathcal{C}_2$
AMVSG10000I10F6	89.93%	23.06%	77.41%	9.83%	83.12%	$\mathcal{C}_2$
AMVSG10000I20F6	80.22%	35.56%	62.89%	20.86%	70.9%	$\mathcal{C}_2$
AMVSG10000I30F6	69.89%	36.09%	66.91%	27.39%	68.27%	$\mathcal{C}_2$

Table 9.33: *Bregman Co-clustering of the AMVSG10000 dataset and related missing-valued variants, without feature clustering. The table shows the results obtained by the best co-clustering basis: the Completeness (Comp.), the Contamination (Cont.), and the Accuracy for each dataset.*

AMVSG10000 - Euclidean - With feature clustering						
Dataset	Star		Galaxy			
	Comp.	Cont.	Comp.	Cont.	Accuracy	Basis
AMVSG10000	100%	7.09%	93.60%	0%	96.52%	$\mathcal{C}_4$
AMVSG10000I5F3	100%	8.56%	92.15%	0%	95.73%	$\mathcal{C}_4$
AMVSG10000I10F3	100%	14.09%	86.25%	0%	92.52%	$\mathcal{C}_4$
AMVSG10000I20F3	99.98%	14.88%	85.35%	0.02%	92.02%	$\mathcal{C}_2$
AMVSG10000I30F3	99.98%	15.46%	84.67%	0.02%	91.65%	$\mathcal{C}_2$
AMVSG10000I5F6	95.72%	6.97%	93.99%	3.67%	94.78%	$\mathcal{C}_4$
AMVSG10000I10F6	90.94%	10.16%	91.39%	7.67%	91.18%	$\mathcal{C}_2$
AMVSG10000I20F6	100%	15.35%	84.8%	0%	91.73%	$\mathcal{C}_4$
AMVSG10000I30F6	99.54%	26.4%	70.07%	0.548%	83.51%	$\mathcal{C}_4$

Table 9.34: *Bregman Co-clustering of the AMVSG10000 dataset and related missing-valued variants, with three feature clusters. The table shows the results obtained by the best co-clustering basis: the Completeness (Comp.), the Contamination (Cont.), and the Accuracy for each dataset.*

those patterns having a magnitude over the 'r' band that assumed values between 18.5 and 21.3. The first two datasets (*Longo 01* and *Longo 02*) were filled with all the objects found in that particular sky slice, whereas in the third one the number of stars and galaxies was hand-tuned in order to have a number of galaxies less than the number of stars. The third dataset (*Longo 03*) was used to show how much the galaxies influence the unsupervised learning process.
From an unsupervised viewpoint these data resulted heavily hard to separate, because the two clusters are strongly overlapped. We use these data to show the different behavior of two techniques with highly nonlinear separable problems.

9.11 Overlapping clusters in Astrophysics: the results

Even though we have only tested the SVC and the Bregman co-clustering (which includes the K-means test), our idea is that the problem at hand cannot be solved

AMVSG5000 - Support Vector Clustering					
Dataset	Star		Galaxy		
	Comp.	Cont.	Comp.	Cont.	Accuracy
AMVSG5000	100%	2.4%	98.37%	0%	99.02%
AMVSG5000I5F3	100%	4.14%	72.43%	0%	83.46%
AMVSG5000I10F3	90.15%	0.5%	99.67%	6.5%	95.86%
AMVSG5000I20F3	100%	13.5%	85.27%	0%	91.16%
AMVSG5000I30F3	100%	15.15%	90.97%	0%	94.58%
AMVSG5000I5F6	95.6%	12.81%	90.63%	3.13%	92.62%
AMVSG5000I10F6	89.1%	18%	89.3%	8.14%	89.22%
AMVSG5000I20F6	79.2%	34.4%	81.8%	16.95%	80.76%
AMVSG5000I30F6	91.25%	11.49%	92.1%	5.95%	91.76%

Details of Support Vector Clustering instances					
Dataset	Kernel	q	C	softening	# runs
AMVSG5000	Gaussian	0.494154	0.002	1	5
AMVSG5000I5F3	Gaussian	0.874066	0.002	1	5
AMVSG5000I10F3	Gaussian	0.609383	0.009	1	6
AMVSG5000I20F3	Gaussian	0.0771601	0.001	1	5
AMVSG5000I30F3	Gaussian	0.0729905	0.0009	1	5
AMVSG5000I5F6	Gaussian	0.402983	0.009	1	4
AMVSG5000I10F6	Gaussian	0.0770513	0.001	1	5
AMVSG5000I20F6	Gaussian	0.380995	0.009	1	5
AMVSG5000I30F6	Gaussian	0.0746409	0.0015	1	6

Table 9.35: *Support Vector Clustering of the AMVSG5000 dataset and related missing-valued variants. The table shows the results obtained by the best co-clustering basis: the Completeness (Comp.), the Contamination (Cont.), and the Accuracy for each dataset. For the last dataset there are no results available due to an unexplained crash of the SVC software running on such data.*

well by "unsupervised classifiers". In confirmation of this, there is the fact that even supervised classifiers need the *a priori* information in order to achieve excellent results (Donalek, 2006, chap. 5). Anyway, we report the results all the same, in order to highlight the behavior of the two studied clustering methodologies with respect to such kind of data.

9.11.1 Bregman Co-clustering

The results of the Euclidean co-clustering about these data are interesting.[34] The true co-clustering process, i.e. with feature compression enabled, always improved the results so far. In the case of the *Longo 01* and *Longo 02* datasets, the compression degraded the quality of the partitioning of some co-clustering basis instead (see Table 9.38, Table 9.39, Table 9.40, and Table 9.41). This is probably due to the intrinsic difficulty of the problem.
According to Table 9.42 and Table 9.43, the reduced number of galaxies with respect to the number of stars allows the co-clustering instances to better separate

[34] As already happened in the previous section, the Information-Theoretic co-clustering was not employed on these data because they are negative-valued.

AMVSG10000 - Support Vector Clustering					
Dataset	Star		Galaxy		
	Comp.	Cont.	Comp.	Cont.	Accuracy
AMVSG10000	99.76%	0.24%	99.8%	0%	99.89%
AMVSG10000I5F3	98.77%	3.29%	97.28%	1.06%	97.96%
AMVSG10000I10F3	89.91%	4.73%	96.44%	8.77%	93.46%
AMVSG10000I20F3	79.96%	4.95%	96.51%	14.82%	88.96%
AMVSG10000I30F3	100%	15.15%	85.04%	0%	91.86%
AMVSG10000I5F6	100%	1.38%	98.82%	0%	99.36%
AMVSG10000I10F6	100%	2.48%	97.87%	0%	98.84%
AMVSG10000I20F6	100%	18.99%	80.35%	0%	89.31%
AMVSG10000I30F6	n/a	n/a	n/a	n/a	n/a

Details of Support Vector Clustering instances					
Dataset	Kernel	q	C	softening	# runs
AMVSG10000	Gaussian	0.0547795	0.001	1	3
AMVSG10000I5F3	Gaussian	0.552747	0.002	1	3
AMVSG10000I10F3	Gaussian	0.552747	0.002	1	3
AMVSG10000I20F3	Gaussian	0.162831	0.001	1	2
AMVSG10000I30F3	Gaussian	0.051751	0.001	1	1
AMVSG10000I5F6	Gaussian	0.757622	0.002	1	3
AMVSG10000I10F6	Gaussian	0.528222	0.001	1	4
AMVSG10000I20F6	Gaussian	0.167996	0.001	1	2
AMVSG10000I30F6	n/a	n/a	n/a	n/a	n/a

Table 9.36: *Support Vector Clustering of the AMVSG10000 dataset and related missing-valued variants. The table shows the results obtained by the best co-clustering basis: the Completeness (Comp.), the Contamination (Cont.), and the Accuracy for each dataset. For the last dataset there are no results available due to an unexplained crash of the SVC software running on such data.*

the two classes of the *Longo 03* data.

9.11.2 Support Vector Clustering

The SVC showed to suffer the negative influence of the galaxies much more than the Bregman co-clustering. In fact, the results about the hardest problems, i.e. the *Longo 01* and *Longo 02* datasets, are the worst results obtained by SVC over the whole experimental phase (see Table 9.44 and Table 9.45). On the contrary, the SVC behaved better than the Euclidean co-clustering on the third Longo dataset, showing a greater performance gain with respect to the reduction of the number of galaxies (see Table 9.46).

9.11.3 Conclusion

The problem at hand cannot be solved well in an unsupervised manner. This is a kind of problem that is suitable for the supervised classifiers. In confirmation of this, we prepared Appendix C in order to present the results of the SVM classifiers about the same data used here.

Longo 01 Data - Euclidean - No feature clustering				
Basis	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_{1,2,4}$	**67.802**%	**38.976**%	**81.534**%	**14.412**%
$\mathcal{C}_3$	49.744%	70.259%	49.891%	30.047%
$\mathcal{C}_5$	49.642%	70.204%	50.123%	29.992%
$\mathcal{C}_6$	63.85%	43.639%	78.919%	16.341%

Longo 01 Data - Euclidean - No feature clustering				
	$\mathcal{C}_{1,2,4}$	$\mathcal{C}_3$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	**77.429**%	49.847%	49.98%	74.414%

Table 9.38: *Bregman Co-clustering of the Longo 01 data, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

Longo 01 Data - Euclidean - With feature clustering				
Basis	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_1$	**67.802**%	**38.976**%	**81.534**%	**14.412**%
$\mathcal{C}_2$	65.043%	41.692%	80.169%	15.678%
$\mathcal{C}_3$	63.884%	43.541%	78.992%	16.315%
$\mathcal{C}_4$	65.145%	41.672%	80.154%	15.642%
$\mathcal{C}_5$	63.884%	43.439%	79.079%	16.3%
$\mathcal{C}_6$	67.973%	50.655%	70.246%	16.277%

Longo 01 Data - Euclidean - With feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	**77.429**%	75.647%	74.475%	75.667%	74.537%	69.566%

Table 9.39: *Bregman Co-clustering of the Longo 01 data, with three feature clusters. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

Anyway, the results of the co-clustering about the most hard datasets (*Longo 01* and *Longo 02*) resulted better than the SVC results (see Table 9.38, Table 9.39, Table 9.40, Table 9.41, Table 9.44, and Table 9.45), whereas the SVC performed best on the third Longo dataset (see Table 9.42, Table 9.43, and Table 9.46).

Dataset	**# of items** (n)	**# dimensions** (d)	**# of classes**
CLASSIC3	3893	3303	3
SCI3	2968	9456	3
PORE	1403	13281	2

Table 9.47: *Summary of datasets used in the text clustering experiments.*

Longo 02 Data - Euclidean - No feature clustering				
Basis	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_{1,2,4}$	**70.819**%	**42.784**%	**80.203**%	**11.98**%
$\mathcal{C}_3$	49.563%	72.323%	51.564%	26.788%
$\mathcal{C}_5$	49.228%	71.317%	54.227%	25.939%
$\mathcal{C}_6$	55.675%	55.43%	74.099%	18.296%

Longo 02 Data - Euclidean - No feature clustering				
	$\mathcal{C}_{1,2,4}$	$\mathcal{C}_3$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	**77.62**%	51.001%	52.847%	69.058%

Table 9.40: *Bregman Co-clustering of the Longo 02 data, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

Longo 02 Data - Euclidean - With feature clustering				
Basis	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_1$	**70.819**%	**42.784**%	**80.203**%	**11.98**%
$\mathcal{C}_2$	61.921%	50.853%	76.033%	15.79%
$\mathcal{C}_3$	55.91%	55.386%	74.036%	18.23%
$\mathcal{C}_4$	62.223%	50.744%	76.021%	15.687%
$\mathcal{C}_5$	56.078%	55.023%	74.337%	18.113%
$\mathcal{C}_6$	58.126%	70.77%	47.368%	24.851%

Longo 02 Data - Euclidean - With feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	**77.62**%	72.165%	69.076%	72.238%	69.341%	50.279%

Table 9.41: *Bregman Co-clustering of the Longo 02 data, with three feature clusters. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

9.12 Clustering in Text Mining: the data

This experimental stage is mainly intended for verifying the ability of the algorithms at hand to deal with high dimensional data. In the second place, we also verify the applicability of such algorithms to the text mining application domain. Actually, the Bregman co-clustering has already showed to be suitable to text mining problems (Banerjee et al., 2007; Dhillon and Guan, 2003a,b; Dhillon et al., 2003b). There are no previous experiments with SVC on text data.
In addition, these experiments may also provide additional informations about the sparse data handling ability of both algorithms. In fact, thousands of words are usually used as features,[35] but they are selected with respect to the whole set

[35]The value of a feature is usually expressed with the TF-IDF weighting system. The Term Frequency (TF) of a term t is simply the number of occurrences of t in a document. Alternatively,

Longo 03 Data - Euclidean - No feature clustering				
Basis	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_{1,2,4}$	**89.8**%	**5.424**%	**79.4**%	**33.943**%
$\mathcal{C}_3$	49.8%	19.417%	52%	79.43%
$\mathcal{C}_5$	49.35%	18.631%	54.8%	78.71%
$\mathcal{C}_6$	99.1%	17.657%	15%	19.355%

Longo 03 Data - Euclidean - No feature clustering				
	$\mathcal{C}_{1,2,4}$	$\mathcal{C}_3$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	**87.72**%	50.24%	50.44%	82.28%

Table 9.42: *Bregman Co-clustering of the Longo 03 data, without feature clustering. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

Longo 03 Data - Euclidean - With feature clustering				
Basis	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
$\mathcal{C}_1$	**89.8**%	**5.424**%	**79.4**%	**33.943**%
$\mathcal{C}_2$	34.65%	39.687%	8.8%	96.743%
$\mathcal{C}_3$	51.6%	22.172%	41.2%	82.453%
$\mathcal{C}_4$	99.15%	17.581%	15.4%	18.085%
$\mathcal{C}_5$	62.45%	11.731%	66.8%	69.217%
$\mathcal{C}_6$	51.25%	16.395%	59.8%	76.531%

Longo 03 Data - Euclidean - With feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	**87.72**%	29.48%	49.52%	82.4%	63.32%	52.96%

Table 9.43: *Bregman Co-clustering of the Longo 03 data, with three feature clusters. The first table shows the Completeness and the Contamination for each class and for each co-clustering basis. The second one shows the Accuracy. The best results are highlighted in bold.*

of documents. Therefore large subsets of features are "not defined" for a particular document, i.e. each document contains only a subset of the whole features set. This characteristic implies the sparseness of the term-document matrix that represents a set of text documents.

the log-TF of t can be used, that is just the natural logarithm of the TF. Used by itself, the TF is the most simple form of the weighting scheme. The Document Frequency (DF) can be used to attenuate the effect of terms that occur too often in the collection to be meaningful for relevance determination. The DF of a term t is the number of documents in the collection that contain that term. The DF cannot be used as it is to scale the term weight. The Inverse Document Frequency (IDF) is calculated instead. The IDF of a term t is

$$\text{IDF}(t) = \log \frac{n}{\text{DF}(t)}.$$

Therefore, the IDF of a rare term is high, whereas the IDF of a frequent term is likely to be low (Manning et al., 2007). The TF-IDF weighting system is simply the product of the TF and IDF.

Longo 01 Data - Support Vector Clustering				
Type	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
GC	**55.06**%	**56.12**%	**86.82**%	**14.12**%

Details of Support Vector Clustering instances						
Type	Kernel	q	C	softening	# runs	Accuracy
GC	Gaussian	0.355449	0.002	1	3	77.33

Table 9.44: *Support Vector Clustering of the Longo 01 data. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy and the details about the SVC instances. The best results are highlighted in bold.*

Longo 02 Data - Support Vector Clustering				
Type	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
GC	**69.816**%	**56.22**%	**76.4**%	**16.1**%

Details of Support Vector Clustering instances						
Type	Kernel	q	C	softening	# runs	Accuracy
GC	Gaussian	0.170214	0.002	1	4	74.43

Table 9.45: *Support Vector Clustering of the Longo 02 data. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy and the details about the SVC instances. The best results are highlighted in bold.*

In the following subsections we present the datasets chosen for this stage. They are among the most common datasets used in the text clustering experiments, so we can compare our results with the ones found in literature.[36] An additional reason to choose these datasets is the poor availability of the term-document matrices of other corpora. In fact, most of the corpora are provided as they are, because the researchers generally preprocess the documents according to their own needs and the characteristics of the learning algorithm to test to.
Unfortunately, due to some reasons[37] we were not able to also engage an *ad hoc* preprocessing phase to build all the datasets for our specific needs. Therefore, we fell back on ready-to-use datasets where it was possible.

[36]The *NG20* and the *CLASSIC3* have already been tested with the co-clustering (Banerjee et al., 2007; Dhillon et al., 2003b).

[37]Mainly, we spent most of our time to verify the properties on several application domains, without specializing the work for a particular domain. Therefore, we did not focus on the preprocessing phase for building the term-document matrices of the chosen corpora.

Longo 03 Data - Support Vector Clustering				
Type	Star		Galaxy	
	Completeness	Contamination	Completeness	Contamination
GC	**90.45**%	**18.2**%	**93.4**%	**30.1**%

Details of Support Vector Clustering instances						
Type	Kernel	q	C	softening	# runs	Accuracy
GC	Gaussian	5.97158×10^{-5}	0.011	1	2	91.04

Table 9.46: *Support Vector Clustering of the Longo 03 data. The first table shows the Completeness and the Contamination for each class and for each SVC instance. The second one shows the Accuracy and the details about the SVC instances. The best results are highlighted in bold.*

9.12.1 CLASSIC3

The *CLASSIC3* data were built over the *SMART* collection from Cornell.[38] The latter consists of MEDLINE, CISI and CRANFIELD sub-collections. MEDLINE consists of 1033 abstracts from medical journals, CISI consists of 1460 abstracts from information retrieval papers and CRANFIELD consists of 1400 abstracts from aerodynamic systems.
The CLASSIC3 dataset was firstly built by Dhillon et al. (2001) for testing a new way to quickly preprocess text documents as bag-of-words. Later, data were used to test the Information-Theoretic Co-clustering (Dhillon et al., 2003b). In the latter, the authors preprocessed the data by removing *stop words*[39] and numeric characters. Afterwards, they selected the top 2000 words by mutual information as feature set.
We found an already preprocessed dataset of the same corpora,[40] where the number of words used as features is 3303 and words were stemmed.[41] The dataset was built by means of the *MC Toolkit*[42] (Dhillon et al., 2001) by selecting the words having a lower bound word frequency of 0.2% and an upper bound word frequency of 15%, i.e. the words that are in less than 0.2% of documents and more than 15% of documents were not used as features.
The way our CLASSIC3 dataset were built is the same followed by Dhillon et al.

[38] Look in `ftp://ftp.cs.cornell.edu/pub/smart`.

[39] Sometimes, some extremely common and semantically non-selective words are entirely excluded from the dictionary, i.e. they are not selected as features. These words are called *stop words*. Examples of typical stop words are the grammatical conjunctions and articles.

[40] Available at `http://www.ise.gmu.edu/~carlotta/teaching/INFS-795-s06/info.html`.

[41] Stemming and lemmatization serve to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form, e.g. car, cars, car's $\Longrightarrow$ car. The two terms differ in their flavor. *Stemming* suggests a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, whereas *lemmatization* suggests doing this properly with the use of a dictionary and morphological analysis of words, normally aiming at returning the base or dictionary form of a word.

[42] Visit `http://www.cs.utexas.edu/users/dml/software/mc/index.html` for getting the software.

(2001), but the number of features is different[43] probably due to the fact that our dataset was built by using the stemmed corpora, whereas the original dataset was constructed by means of a non-stemmed dictionary.

9.12.2 NewsGroup20

The *NewsGroup20 (NG20)* is a collection of 19997 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. To the best of our knowledge, it was originally collected by Lang (1995), though he does not explicitly mention this collection.[44] The NG20 collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering.
The data is organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other (e.g. `comp.sys.ibm.pc.hardware/comp.sys.mac.hardware`), while others are highly unrelated (e.g. `misc.forsale/soc.religion.christian`). Besides, in the original version of corpora, approximately 6% of the articles are cross-posted.[45]
In order to setup easy-to-manage experiments, we chose not to involve all the newsgroups in our tests, as already done in the past by Dhillon et al. (2003b). On the contrary, however, we chose to use all the documents in the selected newsgroups and not only small subsets.
We found a ready-to-use NG20 dataset in bag-of-words format,[46] but the data seemed to be not suitable for the clustering techniques we have to test. Therefore, we chose to perform the preprocessing phase by ourself by using the *MC Toolkit* (Dhillon et al., 2001).[47] We selected our sub-collections from a "duplicate-less" version[48] of the NG20 data, only consisting of 18828 documents.

9.12.2.1 SCI3 sub-collection

The SCI3 sub-collection consist of three newsgroups in the macro-category *science*: `sci.crypt`, `sci.med`, and `sci.space`. The first one is about mathematical/cryptographical topics; the second one treats medical topics; finally, the third newsgroup is about space.
The number of documents in this sub-collection is 2968. The preprocessing phase was the same used to create the CLASSIC3 collection, i.e. the words having a

[43] The original version of the CLASSIC3 data had 4303 features, i.e. 1000 features more.

[44] The collection is currently available at the official home page (`http://people.csail.mit.edu/jrennie/20Newsgroups/`) or at *UCI Knowledge Discovery in Databases Archive* (Hettich and Bay, 1999).

[45] A cross-posted article belongs to more than one newsgroup.

[46] Originally created by Rennie (2001) and available online at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

[47] We used just the features of the MC Toolkit: stop-words filtering and feature selection. Neither stemming nor lemmatization were performed on the sub-collections.

[48] An NG20 version without cross-posted articles. It is available at the official home page (`http://people.csail.mit.edu/jrennie/20Newsgroups/`).

lower bound word frequency of 0.2% and an upper bound word frequency of 15% were selected. The total number of selected words is 9456.

9.12.2.2 PORE sub-collection

The name PORE stands for "POlitics and REligion", and it comes from the newsgroups used to build this sub-collection: `talk.politics.misc` and `talk.religion.misc`. The first one concerns generic discussions about the politics, and so the second one about the religion. At first sight, these two topics could seem to be totally unrelated, but if we take a look to the debates all around the world we see that that first impression is not the right one and the results about these data will confirm this empirical observation that anyone can do.
We have 1403 textual documents in this collection represented by 13281 features. The preprocessing phase was slightly different in this case: the upper bound word frequency was changed to 25%.

9.13 Clustering in Text Mining: the results

In this section we are going to present the results obtained within the text mining application domain by using both the SVC and the Bregman Co-clustering. The behavior of both methodologies was quite expected, for several reasons. In the first place, text mining experiments with the Bregman Co-clustering were already successfully performed in the past (Banerjee et al., 2007; Dhillon et al., 2003b). In the second place, even though no SVC experiments about text data are available,[49] there is a hypothesized difficulty to handle high dimensional data which is already partially proved with the experiment in subsection 9.5.6. Therefore, we expected good results from the former and less good results from the latter.

9.13.1 CLASSIC3

These data are relative simple to separate because the three classes are mostly unrelated. Anyway we will see that good results will be achieved only by the co-clustering and particular instances of the SVC. The classic one way clustering yielded very poor results in the majority of cases and the classic Gaussian SVC had also troubles for separating one class from the other two.

9.13.1.1 Bregman Co-clustering

The results in Table 9.48 show that the classic one-way clustering does not behave well on these data. The only exception is the scheme $\mathcal{C}_6$ with the Euclidean divergence that perfectly separate the three clusters.
The point is different when we enable the feature compression. The number of contextual requested feature clusters is 10. According to the results showed in Table 9.49, the feature compression boosted the co-clustering results in many cases.

[49] To the best of our knowledge.

CLASSIC3 - Euclidean - No feature clustering									
Basis	CISI			MEDLINE			CRANFIELD		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_1$	72.47%	53.01%	61.24%	25.31%	43.27%	31.94%	39.09%	29.57%	33.68%
$\mathcal{C}_{2,4}$	72.5%	53.08%	61.28%	25.41%	43.47%	32.08%	39.17%	29.57%	33.7%
$\mathcal{C}_3$	43.13%	87.67%	57.82%	55%	19.17%	28.44%	30.09%	12.14%	17.3%
$\mathcal{C}_5$	37.6%	37.26%	37.42%	29.05%	33.88%	31.28%	36.5%	32.36%	34.3%
$\mathcal{C}_6$	**100**%	**100**%	**100**%	**100**%	**100**%	**100**%	**100**%	**100**%	**100**%
CLASSIC3 - Information-Theoretic - No feature clustering									
Basis	CISI			MEDLINE			CRANFIELD		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_1$	37.77%	33.42%	35.46%	24.77%	31.46%	27.72%	37.39%	34.43%	35.84%
$\mathcal{C}_2$	38.14%	32.6%	35.16%	26.82%	34.27%	30.1%	36.45%	34.5%	35.44%
$\mathcal{C}_3$	36.87%	31.64%	34.06%	24.48%	30.59%	27.2%	36.03%	34.71%	35.36%
$\mathcal{C}_4$	36.48%	31.23%	33.66%	27.34%	34.56%	30.52%	36.95%	35.29%	36.1%
$\mathcal{C}_5$	37.74%	33.22%	35.34%	28.66%	37.75%	32.58%	35.93%	32%	33.86%
$\mathcal{C}_6$	36.76%	34.52%	35.6%	27.92%	33.3%	30.38%	37.6%	34.64%	36.06%

CLASSIC3 - Euclidean - No feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	41.998%	49.782%	42.332%	42.076%	34.601%	**100%**
Macroaveraging	42.282%	40.643%	34.515%	42.354%	34.337%	**100**%
CLASSIC3 - Information-Theoretic - No feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	33.265%	33.727%	32.469%	33.573%	33.984%	34.241%
Macroaveraging	33.011%	33.565%	32.203%	33.426%	33.923%	34.014%

Table 9.48: *Bregman Co-clustering of the CLASSIC3, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. The Euclidean $\mathcal{C}_6$ scheme yields the best results.*

The schemes from 2 through 5 of the Euclidean co-clustering achieve the perfect separation all. The Information-Theoretic co-clustering improved only the basis $\mathcal{C}_5$, that is also the scheme originally developed in Dhillon et al. (2003b).

9.13.1.2 Support Vector Clustering

The classic SVC with the Gaussian kernel had no trouble to separate the CISI documents from the MEDLINE ones, with 100% of precision/recall. However, the documents belonging to the CRANFIELD sub-collection were not separated from the ones belonging to the other two sub-collections; in fact, 796 of them were assigned to the same cluster where all CISI documents were, whereas the remaining 604 CRANFIELD documents were assigned to the cluster containing all the MEDLINE documents too.
Before trying a different kernel, we have also tried different values for the soft margin parameter other than the one estimated thanks to our heuristics. We also tried the softening strategy to be sure that no useful kernel width values were skipped. This first phase of hand-tuning did not yield different results, therefore

CLASSIC3 - Euclidean - With feature clustering									
Basis	CISI			MEDLINE			CRANFIELD		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_1$	31.88%	39.52%	35.3%	50.8%	49.27%	50.02%	22.66%	17.5%	19.74%
$\mathcal{C}_{2-6}$	**100**%	**100**%	**100**%	**100**%	**100**%	**100**%	**100**%	**100**%	**100**%

CLASSIC3 - Information-Theoretic - With feature clustering									
Basis	CISI			MEDLINE			CRANFIELD		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_1$	39.16%	33.15%	35.9%	28.06%	36.21%	31.62%	38.9%	36.79%	37.82%
$\mathcal{C}_2$	35.61%	31.37%	33.36%	26.72%	34.66%	30.18%	37.25%	33.71%	35.4%
$\mathcal{C}_3$	37.59%	31.64%	34.36%	25.08%	31.85%	28.06%	36.76%	35.5%	36.12%
$\mathcal{C}_4$	36.5%	32.12%	34.18%	25.24%	31.17%	27.9%	35.96%	34.21%	35.06%
$\mathcal{C}_5$	**99.86**%	**100**%	**99.92**%	**99.81**%	**99.9**%	**99.86**%	**99.93**%	**99.71**%	**99.82**%
$\mathcal{C}_6$	37.9%	33.56%	35.6%	26.04%	32.14%	28.78%	35.7%	33.79%	34.72%

CLASSIC3 - Euclidean - With feature clustering		
	$\mathcal{C}_1$	$\mathcal{C}_{2-6}$
Accuracy	34.19%	**100%**
Macroaveraging	35.021%	**100%**

CLASSIC3 - Information-Theoretic - With feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	35.268%	33.085%	33.085%	32.623%	**99.872**%	33.265%
Macroaveraging	35.111%	32.975%	32.847%	32.376%	**99.869**%	33.028%

Table 9.49: *Bregman Co-clustering of the CLASSIC3, with 10 feature clusters. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. Best results are in bold.*

we tried different kernels (see Table 9.50).
The Laplacian kernel performed the worst and did not separate any cluster, i.e. it yielded a single meaningless cluster with any parameters setup. The Exponential kernel worked very well instead. The SVC ran with such a kernel and with a soft margin parameter value estimated by means of our heuristics. Such an SVC instance yielded six clusters. Three of them can be neglected because they are low-cardinality clusters,[50] while the other three clusters correctly collected the most of the documents at hand. Therefore the results are not as "less good" as supposed.
This is a first proof of the applicability of this clustering algorithm to the text mining application domain. Furthermore, once again we provide an evidence of the fact that the SVC together with the CCL can deal with high dimensional data. This contradicts the conjecture about the inability of the SVC to deal with high dimensional data. Actually, as we have already stated in the previous chapters, the ability of the SVC of handling such kind of data depends on the *cluster labeling* stage rather than the *cluster description* one. In our case the cluster assignment is made by means of the CCL that has already proved to be robust with respect to datasets with hundreds of features (Lee and Daniels, 2006). This experiment is a

[50] Two points, three points, and three points respectively.

CLASSIC3 - Support Vector Clustering									
Type	CISI			MEDLINE			CRANFIELD		
	P	R	F1	P	R	F1	P	R	F1
LC	no separation								
GC	64.72%	100%	78.58%	63.11%	100%	77.38%	n.a.n.		
EC	**100**%	**99.8**%	**99.9**%	**100**%	**100**%	**100**%	**100**%	**99.6**%	**99.8**%

	LC	GC	EC
Accuracy	37.50%	64.038%	**99.8**%
Macro-AVG	n/a	n/a	**99.9**%

Details of Support Vector Clustering instances					
Type	Kernel	q	C	softening	# of runs
LC	Laplacian	any	any	any	any
GC	Gaussian	0.527325	0.00256871	any	3
EC	**Exponential**	**1.71498**	**0.00256871**	**1**	**4**

Table 9.50: *Support Vector Clustering of the CLASSIC3. The first table shows the Precision (P), Recall (R), F1, Accuracy and Macroaveraging for each class and for each SVC instance. The second table shows the details about the SVC instances. In bold the best results.*

first proof that it is also able to deal with datasets with thousands of features.

9.13.2 SCI3

These data are not as simple as the CLASSIC3 data. A subset of the documents vocabulary is shared among the three newsgroups, i.e. the three classes are not totally unrelated. Moreover, the features of this dataset are definitely more than the ones used in CLASSIC3.
The aforementioned characteristics were enough to prevent the SVC from solving the problem. We tried a number of parameter settings and kernel combinations, but we did not success in separating the SCI3 data, always obtaining a single all-inclusive cluster. We also tried to use a compressed version of the matrix as already done in subsection 9.5.6, but nothing changed.[51]
The result was different for the Bregman co-clustering. It still showed its ability to work well in high dimensional spaces, even though the clusters are quite overlapped. Unfortunately, for this experiment the results are available only for some approximation schemes:[52] the bases $\mathcal{C}_2$ and $\mathcal{C}_6$ for the Euclidean distance, and the basis $\mathcal{C}_5$ for the I-divergence, corresponding to the works in Cho et al. (2004a) and Dhillon et al. (2003b) respectively.
Without feature clustering (see Table 9.51), the scheme $\mathcal{C}_2$ of the Euclidean co-clustering, i.e. the K-means, showed poor performance, as already happened withe CLASSIC3 data. On the contrary, the $\mathcal{C}_6$ resulted very accurate once again,

[51] Therefore, no tables with SVC results are provided here.

[52] Due to the lack of time, we employed the *Co-cluster* software instead of the *Bregman Co-cluster* software in this experiment, because the former is faster, especially when the matrix has thousands of columns (the features). We will do the same in the next experiment.

SCI3 - Euclidean - No feature clustering									
Basis	SCI.CRYPT			SCI.MED			SCI.SPACE		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_2$	41.05%	51.87%	45.83%	37.71%	63.84%	47.41%	40%	1.6%	3.2%
$\mathcal{C}_6$	**99.88**%	**86.58**%	**92.76**%	**80.11**%	**91.92**%	**85.61**%	**90.75**%	**89.46**%	**90.1**%

SCI3 - Information-Theoretic - No feature clustering									
Basis	SCI.CRYPT			SCI.MED			SCI.SPACE		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_5$	33.39%	98.39%	49.86%	31.92%	1.52%	2.9%	0%	0%	n.a.n.%

SCI3 - Euclidean - No feature clustering		
	$\mathcal{C}_2$	$\mathcal{C}_6$
Accuracy	39.15%	**89.32**%
Macro-AVG	32.12%	**89.5**%
SCI3 - Information-Theoretic - No feature clustering		
	$\mathcal{C}_5$	
Accuracy	33.36%	
Macro-AVG	17.58%	

Table 9.51: *Bregman Co-clustering of the SCI3, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. In bold the best results.*

even without feature clustering enabled. The $\mathcal{C}_5$ with I-divergence also did not behave well without feature clustering.
By enabling the feature clustering (see Table 9.52), the results the Information-theoretic instance were improved, whereas the Euclidean co-clustering with the $\mathcal{C}_6$ scheme got worst. The Euclidean co-clustering with the scheme $\mathcal{C}_2$ gain an irrelevant enhancement.

9.13.3 PORE

The trend of this experiment is the same of the previous one. The SVC failed in every case, whereas the Bregman co-clustering still behave well and in a similar way it did with SCI3 collection (see Table 9.53 and Table 9.54).

9.13.4 Conclusion

We performed the experiments on text data for two main reasons: in the fist place, we want to analyze the behavior of both clustering methods with respect to very high dimensional data. In the second place, we want to determine the applicability of the SVC and the Bregman co-clustering to the text mining application domain.
As far as the Bregman co-clustering is concerned, we know from previous experiments that this technique fits well the application domain at hand. However, our experiments move slightly away from ones performed by Dhillon and Guan (2003a), Dhillon et al. (2003b) or Banerjee et al. (2007). In the majority of our ex-

SCI3 - Euclidean - With feature clustering									
Basis	SCI.CRYPT			SCI.MED			SCI.SPACE		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_2$	54.2%	65.19%	59.18%	49.20%	34.24%	50.8%	38.73%	42.66%	40.6%
$\mathcal{C}_6$	72.43%	56.20%	63.3%	25.7%	13.94%	18.08%	43.26%	72.85%	54.28%
SCI3 - Information-Theoretic - With feature clustering									
Basis	SCI.CRYPT			SCI.MED			SCI.SPACE		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_5$	**92.45**%	**92.63**%	**92.54**%	**87.68**%	**84.85**%	**86.24**%	**88.0**%	**90.68**%	**89.32**%

SCI3 - Euclidean - With feature clustering		
	$\mathcal{C}_2$	$\mathcal{C}_6$
Accuracy	47.37%	47.64%
Macro-AVG	46.72%	45.22%
SCI3 - Information-Theoretic - With feature clustering		
	$\mathcal{C}_5$	
Accuracy	**89.39**%	
Macro-AVG	**89.37**%	

Table 9.52: *Bregman Co-clustering of the SCI3, with 20 feature clusters. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. In bold the best results.*

periments, the best results were yielded by the Euclidean co-clustering by means of the approximation scheme $\mathcal{C}_6$, notwithstanding the Information-Theoretic was supposed to be the best one on textual data. Moreover, the results obtained on the CLASSIC3 dataset are definitely better than the ones obtained in the aforementioned works. Even the SVC results on these data are on the verge of the perfection.

The main difference between our CLASSIC3 co-clustering experiment and the ones in the aforesaid works is that we did not use any improved cluster update strategy or enhanced cluster initialization strategy (see section 9.2). In Dhillon and Guan (2003a), the *local search* strategy[53] improved the Information-Theoretic co-clustering results.

The point is different for the SVC. To the best of our knowledge, no previous SVC experiments were undertaken over the text mining application domain.[54] The first experiment on CLASSIC3 data yielded very good results, showing the ability of the Cone Cluster Labeling (CCL) to deal with thousands of features.

However, the SVC failed on SCI3 and PORE data. The reasons could be a number. Above all, the SCI3 dataset dimensionality is about three times of the CLASSIC3 dataset dimensionality, and the PORE dimensionality is about four times. Hence, it is likely that we found a dimensionality limit for the CCL.

It is also important to recall that the CCL is not the only promising cluster labeling

[53] Also known as *first variation* (Dhillon et al., 2002).

[54] There is only a very recent paper wherein authors make use of a two-phase approach: a first phase employing the pseudo-hierarchical clustering of the SVC, plus a second stage with supervised SVM classifiers (Hao et al., 2007).

PORE - Euclidean - No feature clustering						
Basis	POLITICS			RELIGION		
	P	R	F1	P	R	F1
$\mathcal{C}_2$	51%	52.65%	51.81%	39.138%	37.58%	38.34%
$\mathcal{C}_6$	**68.31**%	**68.13**%	**68.22**%	**60.79**%	**60.99**%	**60.89**%

PORE - Euclidean - No feature clustering		
	$\mathcal{C}_2$	$\mathcal{C}_6$
Accuracy	45.91%	**64.93**%
Macroaveraging	45.1%	**64.55**%

Table 9.53: *Bregman Co-clustering of the PORE, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis (there are no results for Information-Theoretic instances because it did not separate the clusters). The second one shows the Accuracy and the Macroaveraging. In bold the best results.*

approach for the SVC.[55] The other fascinating algorithm is the second generation of the Gradient Descent (GD) (see subsection 6.3.4): it promises very high accuracy in assigning the points to the right clusters. Furthermore, the theoretical fundamentals of the GD suggest that it is more robust than the CCL with respect the high dimensionality.
Moreover the SCI3 and PORE datasets were built by ourself and neither stemming or lemmatization were applied. We used just the *MC Toolkit* feature selection on the whole dictionary. Therefore, it is likely that the construction of these two datasets could be improved (also employing a more sophisticated feature selection strategy) so that the SVC with the CCL could separate them.[56]
However, it worths to spend resources and time to better explore the applicability of the SVC to the text clustering: its ability to deal with high dimensional[57] and sparse data are two important peculiarities necessary to a clustering algorithm to work in text mining application domain. Furthermore, due to its pseudo-hierarchical behavior, the SVC could also be employed for hierarchical automatic organization of text documents, a frequent task in such application domain.[58]

9.13.4.1 More results on sparse data

As already anticipated in subsection 9.9.3, the text clustering experiments give us also some additional informations about the ability of the algorithms to handle sparse data. We recall that with sparse data we indicate data represented by a sparse matrix. The term-document matrices are usually very sparse, therefore a

[55] Let us call in mind that the CCL is the only reliable cluster labeling algorithm available in our SVC software. We chose to implement it first, because of the good tradeoff between computational complexity and clustering performance.

[56] In addition, the results of the Bregman co-clustering could be improved too.

[57] This capability can be improved.

[58] We have only to think to the taxonomy of the news: we have a macro-categorization (e.g. Sport, Politics, Science, etc.) and, for each macro-category, a new categorization (e.g. Basketball, Soccer, Volleyball for the macro-category Sport).

PORE - Euclidean - With feature clustering						
Basis	POLITICS			RELIGION		
	P	R	F1	P	R	F1
$\mathcal{C}_2$	62.06%	49.81%	55.26%	50.19%	62.42%	55.64%
$\mathcal{C}_6$	**80.23**%	**91.1**%	**85.32**%	**86.81**%	**72.3**%	**78.9**%
PORE - Information-Theoretic - With feature clustering						
Basis	POLITICS			RELIGION		
	P	R	F1	P	R	F1
$\mathcal{C}_5$	61.28%	54.32%	57.59%	50.56%	57.64%	53.87%

PORE - Euclidean - With feature clustering		
	$\mathcal{C}_2$	$\mathcal{C}_6$
Accuracy	55.45%	**82.68**%
Macro-AVG	55.45%	**82.11**%
PORE - Information-Theoretic - With feature clustering		
	$\mathcal{C}_5$	
Accuracy	55.81%	
Macro-AVG	55.73%	

Table 9.54: *Bregman Co-clustering of the PORE, with 20 feature clusters. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. In bold the best results.*

good result on such data suggests also a good ability in treating sparse data.
The main difference between the term-document matrices and the matrices representing the astrophysics data is that in the former the "zero" values actually mean that a given word is not in a given document. In the astrophysics, the matrices carry a real absence of information. They are then treated as sparse matrices replacing the empty cells with zero values.
Therefore, the results of our text clustering experiments complete the picture about the data sparseness handling, providing a different, additional, information with respect to the datum given by astrophysics missing values experiments.

9.14 Conclusion

The experiments[59] have been organized in five stages. Several datasets were involved, from different application domains

- 6 miscellaneous datasets in the *preliminary experimental stage*

[59] All the experiments were primarily intended for verifying whether the two clustering algorithms investigated in this thesis were able to address the issues mentioned in chapter 3. In fact, we did not present any result about the specific behavior of the co-clustering paradigm, i.e. we did not show any datum about the feature clusters, because such results are not our primary goal. We use the co-clustering paradigm to address the aforementioned issues instead. Co-clustering-centric results are available in a number of publications (Banerjee et al., 2007; Cheng and Church, 2000; Cho et al., 2004a; Deodhar et al., 2007a; Dhillon and Guan, 2003a; Dhillon, 2001; Dhillon et al., 2003b; Guan et al., 2005; Hartigan, 1972; Long et al., 2005; Tchagang and Tewfik, 2005; Yang et al., 2003).

- 2 synthetic datasets in the *outliers handling stage*
- 2 MVSG datasets for the first session of the *missing values handling stage*
- 18 AMVSG datasets for the second session of the *missing values handling stage*
- 3 datasets for the *unsupervised star/galaxy separation stage*
- 3 datasets for the *text clustering stage*

Totally, 34 datasets were used during the whole experimental phase. If we consider each approximation scheme and each divergence for the Bregman co-clustering, we performed 24 Bregman co-clustering tests[60] for each dataset. Averagely, a couple of tests was also performed with the SVC for each dataset. Therefore, we performed about 832 tests on the first 32 datasets. The remaining two datasets are the last two used in the text clustering stage. We used the *Co-cluster* software in those experiments, which provides only two approximation schemes for the Euclidean distance and only one for the I-divergence. Therefore, 6 Bregman co-clustering tests were performed for each of the last two datasets. At least 5 tests were also performed with SVC for each of the last two datasets, though the results are reported as "failed". Therefore, the total number of tests is about 854.
In the *preliminary experimental stage* we verified a number of properties. Above all, we provided a first evidence of the general applicability of both algorithms. As already stated in the previous chapters, both the Bregman co-clustering framework and the SVC aim at providing us with a general clustering solution which is applicable to a variety of application domain. We also proved the ability of the Bregman co-clustering of improving clustering results and dealing with high dimensional data by means of the contextual feature clustering, one of the peculiarities of this framework. As far as the SVC is concerned, we showed the effectiveness of our contributions to this clustering technique. Moreover, we also proved the ability of this clustering algorithm to deal with overlapping clusters and high dimensional data. Finally, we proved the effectiveness of employing different kernels in some particular experiments.
With the subsequent experimental stage, we verified the hypothesized capability of the SVC to handle very well the data with many outliers. The SVC is currently the first and the unique clustering algorithm that exhibits this characteristic.[61]
With the third experimental stage we performed a number of tests on the missing values robustness of the dataset. We recall that we faced such a problem as a "simple" data sparseness problem, i.e. the empty cells were treated as zero values and no imputation or estimation[62] were performed. Also, we call up once again that

[60] 12 tests without feature clustering and 12 with feature clustering.

[61] Recently, the ultimate generalization of the Bregman framework (Deodhar et al., 2007a) introduced a novel characteristic that is unusual for clustering algorithms based on a relocation scheme. The "bubble co-clustering" promises to involve in the clustering process only dense areas determined on local rather than global quality, by pruning away the less informative parts of the data matrix. Since the outliers are typically far away from dense areas, this novel mechanism should be make the Bregman framework robust with respect to the outliers too.

[62] See section 3.7 for details.

the Bregman framework could face this problem also by assigning a non-uniform probability distribution to the matrix elements (therefore, assigning a zero probability to missing values). However, this peculiarity is not fully implemented in the available softwares, therefore the tests with the Bregman co-clustering were performed by exploiting only its contextual feature clustering ability. Actually, facing this problem as a data sparseness problem allowed us to perform a fair comparison between the Bregman co-clustering and the SVC, which provides no explicit way to handle missing values.
Our missing values experiments were carried out within the astrophysics application domain. The first sub-phase was borrowed from previous works carried out at the *CALTECH* (Wagstaff, 2004; Wagstaff and Laidler, 2005). The second subphase are still about the astrophysics data, but this time the missing values were artificially "injected" into the datasets, depriving the data of an increasing quantity of information. These missing values experiments within the astrophysics application domain are the first performed with Bregman co-clustering and the SVC, and the results are very promising. Furthermore, this type of tests about missing values were not previously performed. On the contrary, the are some previous works about the ability of the Bregman framework to estimate the missing values (Huynh and Vu, 2007; Waters and Raghavan, 2007).
The second type of experiments within the astrophysics application domain was the unsupervised star-galaxy separation. The experiments shown that the separation of such data is not well accomplished by unsupervised algorithms.[63] The SVM classifiers results in Appendix C provide additional evidences of this fact.
Finally, the last application domain we were interested into is the text clustering. The applicability of the Bregman co-clustering are definitely showed by the three experiments performed here and by the others previously carried out (Banerjee et al., 2007; Dhillon and Guan, 2003a; Dhillon et al., 2003b). Its extraordinary ability to deal with very high dimensional data (up to 12300 features in our case) is the keystone for facing the problem of the text clustering.
Some doubts remain about the SVC. The first experiment on CLASSIC3 yielded unexpectedly optimal results, thanks to the employment of the Exponential kernel instead of the classically suggested Gaussian kernel. Anyway, the CLASSIC3 data are the simplest ones used in our text clustering stage. On the other hand, the other two datasets (SCI3 and PORE) were built by ourself with the strictly necessary instruments. Therefore, we do not actually know neither if building better these data would improve the SVC results nor if we really found the dimensionality limit for the CCL.
However, in both cases it worths to further explore the applicability of the SVC to the text clustering: its ability to deal with high dimensional data can be improved. Moreover, its capability to handle sparse data and its pseudo-hierarchical behavior, would make the SVC very useful in the text clustering application domain.
To conclude, the whole experimental stage showed the superiority of both algorithm over classical algorithms like the K-means. In addition, we isolated the peculiarities of both state-of-the-art clustering techniques analyzed here, showing how they address a number of issues mentioned in chapter 3: missing values,

[63] At least, by the two clustering algorithms considered in this thesis.

data sparseness, high dimensionality, estimation of the number of clusters, outliers handling and robustness with respect to the noise, arbitrary-shaped clusters handling. Furthermore, both algorithms showed the ability to be independent of the application domain. Finally, we have successfully highlighted the effectiveness of our contributions to the SVC.

CHAPTER

TEN

Conclusion and Future Work

«*Gutta cavat lapidem non vi, sed saepe cadendo.*[a]»

[a] The drop excavates the stone, not with force but by falling often.
PUBLIUS OVIDIUS NASO
(NASO, 10)

CLUSTERING is a technique to group a set of objects into subsets or *clusters*. The goal is to create clusters that are coherent internally, but substantially different from each other, on a per-feature basis. Clustering is an unsupervised learning technique, so the cluster membership is determined only by the spatial and statistical characteristics of the data.
We were looking for clustering techniques that were able to deal well with different domains and at the same time were able to address some famous issues, such as nonlinear separable problems, outliers, high dimensionality, missing values, and so on. In this thesis we have investigated two innovative clustering techniques that address such requirements: the *Minimum Bregman Information (MBI) principle for Co-clustering* (or simply *Bregman Co-clustering*) and the *Support Vector Clustering (SVC)*.

10.1 MBI principle for Co-clustering

The Minimum Bregman Information (MBI) principle for Co-clustering (see chapter 5) provides a very powerful generalization of the classical relocation scheme already adopted by famous clustering algorithms like the K-means or Linde-Buzo-Gray (LBG). It generalizes the clustering in two directions. In the first place,

it allows to employ a wide range of the distortion functions, i.e. the Bregman divergences. In such a way it allows to generalize the *Maximum Entropy* principle (based on the KL-divergence), the *Minimum Sum Squared Residue* principle (based on the squared Euclidean distance), and many potentially unknown principles which rely on unknown divergences.
In the second place, it allows to generalize what we want to cluster. In fact, the proposed co-clustering framework is so flexible that enables to clustering only data, only features or both at the same time. Moreover, it provides six schemes which differ to each other for the statistical information they allow to retain.
It also showed that the simultaneous clustering leads to greater improvements of the clustering results with respect to the ones obtained by employing the feature clustering as a preprocessing step of the data clustering.
The Bregman co-clustering also performed well with nonlinear separable problems, notwithstanding it does not employ explicit nonlinear mappings. It can also face well the high dimensionality thanks to the built-in feature clustering step, and it is resulted robust with respect to the missing values and sparse data too. Furthermore, it is independent of the application domain thanks to the ability of employing different distortion functions. In fact, for each type of data we can exploit the peculiarities of a particular Bregman divergence (or develop a new one) that best fit the application domain at hand.
Finally, the Bregman co-clustering can be also used for tasks other than the clustering, such as the matrix approximation and the missing values prediction. In both of these cases it showed very promising performance.
The main drawbacks are the inability to deal with the outliers and the necessity of estimating the number of clusters. However, very recent works have further generalized the Bregman co-clustering framework in order to address such issues (Deodhar et al., 2007a,b).[1]

10.2 Support Vector Clustering

We have detailedly reviewed the Support Vector Clustering (SVC) and its characteristics in chapter 6.[2]
This method resulted a very promising approach to the clustering problem and address most of the issues mentioned in the opening of this chapter as well as in chapter 3. The main characteristics are the ability to deal with clusters of arbitrary shape and automatically discover the cluster structure (and so the number of clusters) thanks to its pseudo-hierarchical behavior. Moreover, SVC is robust to the noise and directly deal with the outliers, putting them in separated low-cardinality (or singleton) clusters. Another key property is the ability to work within different application domains, thanks to the employment of different kernels and its ability to analyze data at different levels of detail. It is also scalable,

[1] We were not able to test such extension due to the too recent publication and lack of publicly available instruments.

[2] There are no other works that contain all the currently available literature about the SVC. Moreover, chapter 6 includes some our own contributions.

high-dimensionality ready and, surprisingly, robust with respect to the sparse data and missing-valued data. Finally, if we employ the GD cluster labeling strategy, we have a unique capability: the ability of extending clusters to enlarged clustered domains which partition the whole data space in many disjoint subdomains.[3] This enable us to decide in what clusters completely unknown points have to be put (see subsection 6.3.4 for details).

10.2.1 Contribution

We have provided some contributions to the SVC, such as a different politic to assign BSVs to clusters with the CCL. We also provided some contributions to the secant-like kernel width generator: a heuristics which allows to reach better results with less iterations in many cases; a modification which drastically reduces the overall running time, avoiding to perform a number of useless computations; finally, we showed that the secant-like kernel width generator can provide kernel width values for any normalized kernel and not only for the Gaussian one.
We have investigated the use of other kernel types and have inferred some basic requirements a kernel has to satisfy to be suitable for the SVC. Finally, we have also presented some new ideas for stopping criteria.
From a practical perspective, we have provided a new modular and extensible software for the SVC, which also includes a set of classes for developing a complete data mining library.

10.3 Future Work

In the case of the Bregman framework, it worths to analyze Bregman divergences other than the *Euclidean* and the *I-divergence* ones, also by developing domain-specific divergences. Moreover, the Bregman Bubble Co-clustering sounds interesting too (Deodhar et al., 2007a,b), and it worths to study it in depth. The Bregman Bubble Co-clustering extends the Bregman framework with outliers handling ability and the auto-detecting of the suitable number of clusters.
As far as the SVC is concerned, one of the issues that still remain open is the robust estimation of the soft margin parameter C.[4] Moreover, further research on cluster labeling would improve the accuracy and the speed of the whole clustering process. Finally, the enhancement of multi-sphere support vector methods for clustering is also a very interesting and still open work.
Anyway, a very captivating work from our perspective is finding a way to connect the explored works.

[3] The number of subdomains corresponds to the number of the clusters.

[4] A promising approach in this direction was presented in Yankov et al. (2007). See also section 6.9.

10.3.1 Connecting the two works

There are essentially two alternative ways to connect the two works. First, we can find a way to combine the two techniques as they are, exploiting the peculiar characteristics of both of them. The more straightforward idea in this direction is to use the co-clustering as a preprocessing step for the SVC. As already stated many times, the high dimensionality is a problem because it makes the clustering process harder both from a computational complexity viewpoint and an accuracy viewpoint, i.e. the clustering takes longer and becomes less accurate. We know that the co-clustering produces a matrix approximation during the clustering process. We can obtain an approximated matrix which has the same number of rows of the original matrix (so, the same number of objects) but a reduced number of features. To do that, we ask the co-clustering to perform only the feature clustering, i.e. to perform the one-way clustering of the transposed matrix. However, this is a simple but limited approach. Furthermore, it is not clear yet whether it is a feasible approach or not. A first experiment[5] in this direction showed its effectiveness, but other experiments[6] did not yield good results.
An interesting alternative is to borrow a "piece" of the one approach to augment the other one. For example we can borrow the kernels from the SVC to use *kernelized* metrics in the Bregman co-clustering, but this may lead to the drawbacks yet summarized in subsection 4.3.2. A way which worths to cover is the utilization of the Bregman divergences for a more general formulation of the Minimum Enclosing Ball (MEB) problem. We recall that the MEB problem is the basis of the all support vector methods for clustering presented in this thesis, therefore this opens out the roads of the research. For example, if we were able to exploit a *Minimum Enclosing Bregman Ball (MEBB)* in the cluster description stage of the SVC, we can also modify the cluster labeling algorithms by means of Bregman divergences.

10.3.1.1 Minimum Enclosing Bregman Ball

Nielsen and Nock (2006) and Nock and Nielsen (2005) have proposed an extension of the Bâdoiu-Clarkson (BC) algorithm for the calculation of the MEB (Bâdoiu and Clarkson, 2003) that generalizes the problem by employing the Bregman divergences instead of the classic Euclidean distance. The only change in the mathematical formulation of the problem is the replacement of the Euclidean distance with a more general Bregman divergence. In addition, much theoretical work is provided to mathematically justify such an extension of the BC algorithm. The authors of such a generalization have observed that in several cases the data are distributed in such a way that it is more useful to enclose them by means of a different envelope. Such an envelope can be obtained finding the *Minimum Enclosing Bregman Ball (MEBB)*, i.e. a MEB which uses a more generic Bregman divergence.
The modification of the BC algorithm still provides the support vector descrip-

[5] The experiment of the Internet advertisement dataset in subsection 9.5.6.
[6] The last two text clustering experiments (see section 9.13).

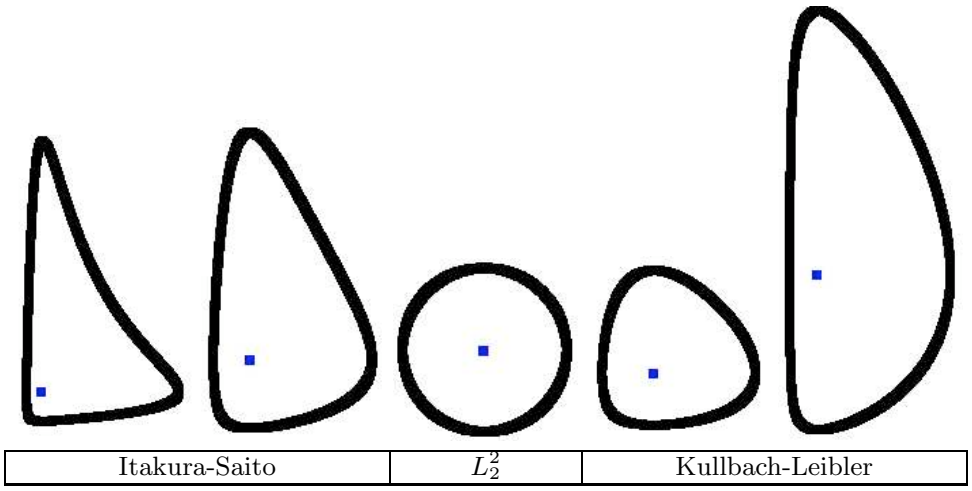


Figure 10.1: *Examples of Bregman balls. The two ones on the left are balls obtained by means of the* Itakura-Saito *distance. The middle one is a classic* Euclidean *ball. The other two are obtained by employing the* Kullback-Leibler *distance (Nock and Nielsen, 2005, fig. 2).*

tion of the data, therefore we can take it into account for much research in the SVC and generally in the SVM. In fact, we wish to recall that the classical BC algorithm is the optimization algorithm exploited by the already mentioned CVMs. The CVMs reformulate the SVMs as a MEB problem and we already expressed our will of testing such machines for the cluster description stage of the SVC (see section 6.12). Since the BC algorithm has been generalized to Bregman divergences, the research about *vector machines* (and therefore about the SVC) could have very interesting implications. We definitely intend to explore this way.

10.3.2 Improve and extend the SVC software

For the sake of accuracy and in order to perform more robust comparisons with other clustering algorithms, an improved and extended software for the Support Vector Clustering (SVC) is needed. More stability and reliability is necessary. Moreover, it is important to implement all the key contributions to this promising technique proposed all around the world. In fact, all the tests have been currently performed by exploiting only some of the characteristic and/or special contribution at time.

APPENDIX

A

One Class classification via Support Vector Machines

«Mathematical reasoning may be regarded rather schematically as the exercise of a combination of two facilities, which we may call intuition and ingenuity.»

ALAN TOURING
(TURING, 1939)

In this appendix we show the equivalence between the two SVM formulations for the *One class classification* problem. We have already discussed the Support Vector Domain Description (SVDD) in the subsection 6.1.1. Here we briefly introduce the *One class SVM* by Schölkopf et al. (2000b), based on the ν-SVM formalism by Schölkopf et al. (2000a).
The proof of this equivalence justifies the employment of the *One class SVM* implementation in LIBSVM as the basis for our SVC implementation described in chapter 8, notwithstanding we used the SVDD formalism in theoretical presentation.
For the sake of completeness, we first introduce the more general ν-SVM formalism that reformulate an SVM classifier from a slightly different viewpoint. Next, we introduce the One Class SVM based upon the latter. Finally, we prove the equivalence between the SVDD and the One Class SVM.

A.1 The ν-Support Vector Machine

The ν-SVM is an alternative SVM formulation for the classification and the regression. It was introduced by Schölkopf et al. (2000a). The primal (nonlinear)

formulation is

$$\min_{\vec{w},b,\xi,\rho} \frac{1}{2}\vec{w}\vec{w} - \nu\rho + \frac{1}{n}\sum_k \xi_k \tag{A.1}$$

subject to

$$y_k[\vec{w}\phi(\vec{x}_k) + b] \geq \rho - \xi_k, k = 1, 2, \cdots, n$$
$$\xi_k \geq 0, \rho \geq 0, k = 1, 2, \cdots, n$$

The dual problem is

$$\min_{\beta} \frac{1}{2}\sum_{kl} \beta_k\beta_l y_k y_l K(\vec{x}_k, \vec{x}_l) \tag{A.2}$$

subject to

$$0 \leq \beta_k \leq 1/n, k = 1, 2, \cdots, n$$
$$0 = \textstyle\sum_k \beta_k y_k, k = 1, 2, \cdots, n$$
$$\textstyle\sum_k \beta_k \geq \nu$$

As in previous chapters, ϕ is a nonlinear mapping from the input space to the feature space, $K(\cdot)$ is a Mercer kernel, n is the dataset cardinality and β_k are the Lagrangian multipliers.
In the ν-SVM formulation, the parameter ρ is the margin, while the parameter ν is an upper bound on the fraction of training errors and a lower bound on the fraction of support vectors.

A.2 The One Class Support Vector Machine

In the SVDD the hypersphere was chosen because it gives a closed boundary around the data. The viewpoint of the One class SVM (or 1-SVM) is different. Here a hyperplane is placed such that it separates the dataset from the origin with maximal margin. The 1-SVM is based of the ν-SVM formalism and the primal (nonlinear) formulation is

$$\min_{\vec{w},\xi,\rho} \frac{1}{2}\vec{w}\vec{w} - \rho + \frac{1}{n\nu}\sum_k \xi_k \tag{A.3}$$

subject to

$$\vec{w}\phi(\vec{x}_k) \geq \rho - \xi_k, k = 1, 2, \cdots, n$$
$$\xi_k \geq 0, k = 1, 2, \cdots, n$$

and the dual problem is

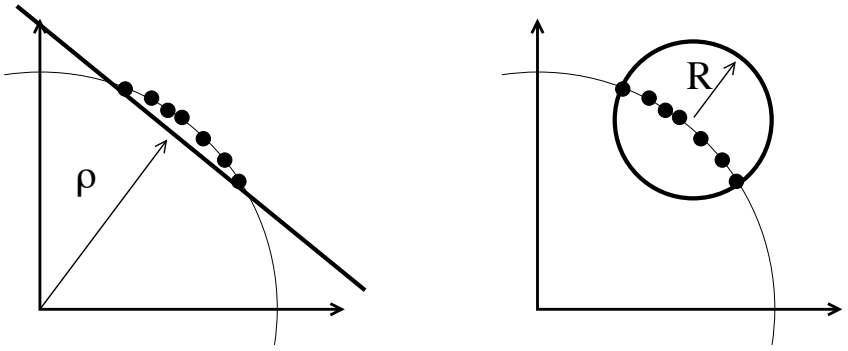


Figure A.1: *Data descriptions by the 1-SVM and the SVDD where the data is normalized to unit norm (Tax, 2001, fig. 2.11).*

$$\min_{\beta} \frac{1}{2} \sum_{kl} \beta_k \beta_l K(\vec{x}_k, \vec{x}_l) \tag{A.4}$$

subject to

$$0 \leq \beta_k \leq 1/n\nu, k = 1, 2, \cdots, n$$
$$1 = \textstyle\sum_k \beta_k, k = 1, 2, \cdots, n$$

A.3 SVDD and 1-SVM equivalence

Although the 1-SVM does not build a closed boundary around the data, it gives identical solutions (i.e. the same Lagrangian multipliers) when the data is scaled to have unit norm.[1]
In view of the necessity of unit-norm scaled data, we can rewrite the 1-SVM primal problem (Equation A.3) in an equivalent form

[1]Incidentally, normalization of a vector is a process in which the components of the vector are divided by norm. Therefore, the norm information is lost by normalization. One approach to solve this problem is to add norm information to the vector before normalization is performed. This could be applying a specific function to the norm and adding the resulting value to the original vector itself before the normalization process. An alternative way to make a vector normalization without loss of information is by adding an extra bias term of 1 to the original vector and then normalize the extended vector

$$\vec{x}' = \frac{(\vec{x}, 1)}{\|(\vec{x}, 1)\|}$$

$$\max_{\xi,\rho} \rho - \frac{1}{n\nu}\sum_{k}\xi_k \tag{A.5}$$

subject to

$$\begin{aligned} \vec{w}\phi(\vec{x}_k) &\geq \rho - \xi_k, k = 1, 2, \cdots, n \\ \xi_k &\geq 0, k = 1, 2, \cdots, n \\ \|\vec{w}\| &= 1 \end{aligned}$$

An extra constraint on $\|\vec{w}\|$ is introduced, and the optimization is now over ρ only. When all data is normalized to unit norm vectors, the equivalence between the SVDD and the hyperplane approach can be shown. Therefore, let us suppose that the input data are normalized data and rewrite the constraints of the SVDD primal problem formulation (Equation 6.2) as follows

$$\begin{aligned} \|\vec{x_k}\|^2 - 2\vec{a}\vec{x}_k + \|\vec{a}\|^2 &\leq R^2 + \xi_k \\ 1 - 2\vec{a}\vec{x}_k + 1 &\leq R^2 + \xi_k \\ \vec{a}\vec{x}_k &\geq \tfrac{1}{2}(2 - R^2 - \xi_k) \end{aligned} \tag{A.6}$$

By changing the signs in Equation 6.2, by introducing an extra constant 2, and in view of Equation A.6, we get

$$\max_{R,\vec{a},\xi}(2 - R^2 - C\sum_{k}\xi_k) \tag{A.7}$$

subject to

$$\vec{a}\vec{x}_k \geq \tfrac{1}{2}(2 - R^2 - \xi_k), k = 1, 2, \cdots, n$$

Now we redefine

$$\vec{w} = \vec{a}, \rho = \frac{1}{2}(2 - R^2), \frac{1}{n\nu} = C, \hat{\xi}_k = \frac{1}{2}\xi_k \tag{A.8}$$

where $k = 1, 2, \cdots, n$. By rewriting the 1-SVM primal formulation using the above definitions, we obtain the 1-SVM formulation equivalent to the SVDD formulation, under the condition that the data must be scaled to the unit norm

$$\max_{\hat{\xi},\rho} 2\left(\rho - \frac{1}{n\nu}\sum_{k}\hat{\xi}_k\right) \tag{A.9}$$

subject to

$$\begin{aligned} \vec{w}\phi(\vec{x}_k) &\geq \rho - \hat{\xi}_k, k = 1, 2, \cdots, n \\ \hat{\xi}_k &\geq 0, k = 1, 2, \cdots, n \end{aligned}$$

which is equal to Equation A.5 up to a factor 2 in the error function.

In the case of a Gaussian kernel, the data is implicitly rescaled. Therefore, the solutions of the SVDD and 1-SVM are identical when the Gaussian kernel width equals q and $C = 1/n\nu$ is used. In their practical implementation the 1-SVM and the SVDD operate comparably. Both perform best when the Gaussian kernel is used. In the SVDD the width parameter is optimized to find a prespecified fraction of support vectors. The parameter C is set at a prespecified value indicating the fraction of objects which should be rejected. In the 1-SVM the ν directly gives the fraction of objects which is rejected. Although it is not specified explicitly, the kernel width in the 1-SVM can be optimized in the same way as in the SVDD.[2]

[2]Our experimental session confirms this, since the SVC software use the 1-SVM instead of the SVDD.

APPENDIX

B

Resource usage of the algorithms

«Premature optimization is the root of all evil (or at least most of it) in programming.»

DONAL ERVIN KNUTH
(KNUTH, 1974, P. 671)

THIS appendix is intended for providing major details about the hardware resources consumption of the algorithms. Such detailed data are available only for the preliminary experiments and for the outliers handling experiments. Anyway, the different nature of each dataset allow us to provide a significant sample for understanding the usage of the resources from the softwares that implement the studied algorithms.
On average, both softwares[1] are little time and space consuming, especially if we take into account that both are experimental softwares that are still not ready for a production environment.

B.1 Support Vector Clustering

For the Support Vector Clustering (SVC) we exclusively provide the statistics about the time taken to complete a clustering process. We choose to not provide memory usage information for each experiment and for each execution because the SVC memory usage was quite constant over all experiments. In fact, the minimum memory usage was about 110MB, whereas the maximum memory usage was about 150MB. It is important to remark that 100MB are always allocated from LIBSVM, because it is the default memory reserved for the kernel matrix cache

[1] Our *damina* software for the SVC and the *Bregman Co-cluster* software.

Support Vector Clustering - CPU usage (in seconds)						
Data	G	GC	GCS	GCS+	ECS	LC
Iris Plant	0.05	0.02	0.01	n/a	0.01	n/a
W. Breast Cancer	n/a	0.17	0.16	0.13	n/a	n/a
Wine Recognition	n/a	0.06	0.04	0.01	n/a	n/a
Quadruped Mammals	n/a	n/a	n/a	n/a	n/a	161
Mushroom	n/a	138.85	n/a	n/a	n/a	n/a
Internet advertisement	n/a	25.37	n/a	n/a	n/a	n/a
Syndeca 02	0.83	n/a	0.58	n/a	n/a	n/a
Syndeca 03	292.04	n/a	127.41	n/a	n/a	n/a

Table B.1: *The Support Vector Clustering (SVC) CPU usage. The time taken by an SVC instance is the sum of the time taken by every single iteration needed to find the most suitable kernel width out. The "n/a" wording indicates that an instance was not used for that experiment. For details about the various instances, refer to the tables in chapter 9.*

and we chose to leave it unchanged. Therefore, the actual memory usage of the remaining parts of the SVC software is very "light".
The data in Table B.1 confirm that SVC does not scale well with respect to the number of features. On the contrary, it behaved well with very large datasets.

B.2 Bregman Co-clustering

In the case of the Bregman Co-clustering the memory usage differs case by case, therefore we detailedly report such informations for each scheme and for each experiment. It is clear from Table B.2, Table B.4, Table B.3 and Table B.5 that a single iteration of a Bregman co-clustering instance is very fast, but the overall time taken by a single instance for accomplishing a clustering process is increased by twenty times. In fact, as reported in section 9.2, each instance was executed twenty times, in order to have results that do not depend on the clusters initialization very much.
The other thing that we note looking at the aforesaid tables is that each scheme has a different running time. Anyway, the latter depends on the computational complexity of the algorithm that implements the approximation scheme and not on the complexity of the approximation scheme itself. In fact, the most CPU-consuming algorithm does not necessarily correspond with the scheme that retains more information about the original matrix.

Euclidean Co-clustering with Feature Clustering - CPU and Memory usage						
Data	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Iris Plant						
Time	0.0071	0.0079	0.0057	0.0076	0.0062	0.007
Memory	14100	14868	180692	23188	195092	357908
W. Breast Cancer						
Time	0.079	0.082	0.085	0.106	0.098	0.222
Memory	95536	97104	960640	118496	1080336	5966080
Wine Recognition						
Time	0.082	0.078	0.073	0.078	0.074	0.081
Memory	48096	56880	549216	102576	579808	1631040
Quadruped						
Time	2509	1058	1178.2	1775.2	1806.4	1650
Memory	4396388	4392292	13769316	4608740	17909988	48580708
Mushroom						
Time	35.9	36.46	37.81	35.71	38.02	50.95
Memory	11000216	11003640	35966520	11319944	34472472	92466840
Internet ads						
Time	242.41	225.85	231.56	995.18	991.32	340.28
Memory	61364336	61365840	70500144	77946800	113707536	102403248
Syndeca 02	n/a					
Syndeca 03	n/a					

Table B.2: *The Euclidean Co-clustering (with feature clustering) CPU and Memory usage. The first is expressed in seconds, whereas the second one in bytes. The time taken by a co-clustering instance is the time of an iteration multiplied by 20.*

Information-Theoretic Co-clustering with Feature Clustering - CPU and Memory usage						
Data	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Iris Plant						
Time	0.014	0.011	0.012	0.009	0.008	0.008
Memory	16788	14996	280532	22548	202868	342932
W. Breast Cancer						
Time	0.037	0.037	0.038	0.038	0.04	0.042
Memory	92016	92496	428976	98736	435216	1108656
Wine Recognition	n/a					
Quadruped	n/a					
Mushroom	n/a					
Internet ads	n/a					
Syndeca 02	n/a					
Syndeca 03	n/a					

Table B.3: *The Information-Theoretic Co-clustering (with feature clustering) CPU and Memory usage. The first is expressed in seconds, whereas the second one in bytes. The time taken by a co-clustering instance is the time of an iteration multiplied by 20.*

Euclidean Co-clustering without Feature Clustering - CPU and Memory usage						
Data	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Iris Plant						
Time	0.0061	0.006	0.0021	0.0063	0.0022	0.006
Memory	12640	11248	82384	18208	84784	185104
W. Breast Cancer						
Time	0.056	0.055	0.024	0.058	0.025	0.066
Memory	92968	91848	427048	104168	432328	897128
Wine Recognition						
Time	0.061	0.062	0.009	0.064	0.0094	0.057
Memory	41720	38776	117688	82136	124408	578488
Quadruped						
Time	1741.3	1736.8	192.1	1919.7	217.2	647.6
Memory	4384376	4380376	6770136	4672376	6805176	10804312
Mushroom						
Time	3.82	3.80	4.03	4.47	4.66	30.35
Memory	10986448	10985968	14886448	11040208	14940688	32494608
Internet ads						
Time	175.884	174.765	28.00	208.50	35.023	792.47
Memory	61358248	61355208	62924968	66094568	63673288	121634456
Syndeca 02						
Time	0.652	0.637	0.0486	0.648	0.050	0.406
Memory	161268	155508	612948	214164	618228	3402868
Syndeca 03						
Time	3.865	3.837	0.169	3.952	0.175	4.24
Memory	503908	494980	5245828	531428	5247748	84802052

Table B.4: *The Euclidean Co-clustering (without feature clustering) CPU and Memory usage. The first is expressed in seconds, whereas the second one in bytes. The time taken by a co-clustering instance is the time of an iteration multiplied by 20.*

Information-Theoretic Co-clustering without Feature Clustering - CPU and Memory usage						
Data	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Iris Plant						
Time	0.012	0.012	0.0026	0.012	0.0027	0.0072
Memory	14704	12624	82384	23024	23024	179984
W. Breast Cancer						
Time	0.029	0.028	0.030	0.029	0.031	0.033
Memory	91048	90568	427048	95848	432328	435688
Wine Recognition	n/a					
Quadruped	n/a					
Mushroom	n/a					
Internet ads	n/a					
Syndeca 02						
Time	0.375	0.362	0.0616	0.376	0.0639	0.507
Memory	143028	140308	612948	167572	618228	3369108
Syndeca 03						
Time	2.951	2.863	0.215	3.116	0.222	4.314
Memory	477700	471844	5245828	496228	5247748	69244804

Table B.5: *The Information-Theoretic Co-clustering (without feature clustering) CPU and Memory usage. The first is expressed in seconds, whereas the second one in bytes. The time taken by a co-clustering instance is the time of an iteration multiplied by 20.*

APPENDIX

C

Star/Galaxy separation via Support Vector Machines

«A galaxy is composed of gas and dust and stars – billions upon billions of stars.»

CARL SAGAN
(UNSOURCED)

THE Star/Galaxy separation problem has been faced in section 9.10 by means of clustering algorithms, i.e. in an unsupervised manner. The experiments have shown that the particular problem represented by *Longo 01*, *Longo 02*, and *Longo 03* datasets cannot be solved at optimum with an "unsupervised classifier". In this appendix we will show the results obtained by SVM classifiers tested on the same data. More specifically, we use three different formalizations of the SVM: the classic Vapnik SVMs (see section 2.6), the Schölkopf ν-SVMs (see section A.1 and references therein), and the Core Vector Machines (CVMs), already mentioned in subsubsection 6.8.1.2 and detailedly described in Tsang et al. (2005), Tsang et al. (2006), Tsang et al. (2007), and Asharaf et al. (2007).

C.1 Training set and Test sets

The training set was built in the same way the *Longo* datasets were (see section 9.10), retrieving the objects from a sky slice that is different from the ones used to fill the *Longo* datasets. It consists of 200 stars and 347 galaxies.
All the SVM implementations were trained with the same training set and then were tested on the *Longo* datasets.

C.2 The results

The results are in Table C.1, Table C.2, and Table C.3.

C.3 The software

For the classic SVM implementation and the ν-SVM implementation, we used the already mentioned LIBSVM (Chang and Lin, 2007). To use the CVMs we employed the tool provided by Tsang et al. (2005).[1]

Longo 01 Data - Classification via SVM					
Type	Star		Galaxy		
	Completeness	Contamination	Completeness	Contamination	Accuracy
SVM	98.27%	6.28%	94.0%	1.64%	97.1%
ν-SVM	**98.33**%	**5.55**%	**94.67**%	**1.61**%	**97.33**%
CVM	98.2%	5.87%	94.4%	1.73%	97.14%

Table C.1: *Classification via Support Vector Machines (SVMs) of the Longo 01 data. The table shows the the Completeness and the Contamination for each class and for each SVM formalization, and also the overall accuracy for each SVM formalization. In bold the best results.*

Longo 02 Data - Classification via SVM					
Type	Star		Galaxy		
	Completeness	Contamination	Completeness	Contamination	Accuracy
SVM	98.6%	4.25%	95.87%	1.4%	97.18%
ν-SVM	**98.4**%	**4.0**%	**96.1**%	**1.6**%	**97.2**%
CVM	98.6%	4.25%	95.87%	1.4%	97.18%

Table C.2: *Classification via Support Vector Machines (SVMs) of the Longo 02 data. The table shows the the Completeness and the Contamination for each class and for each SVM formalization, and also the overall accuracy for each SVM formalization. In bold the best results.*

Longo 03 Data - Classification via SVM					
Type	Star		Galaxy		
	Completeness	Contamination	Completeness	Contamination	Accuracy
SVM	98.5%	4.94%	95.23%	1.48%	97.48%
ν-SVM	**98.44**%	**4.43**%	**95.7**%	**1.51**%	**97.61**%
CVM	98.4%	4.77%	95.4%	1.55%	97.45%

Table C.3: *Classification via Support Vector Machines (SVMs) of the Longo 03 data. The table shows the the Completeness and the Contamination for each class and for each SVM formalization, and also the overall accuracy for each SVM formalization. In bold the best results.*

[1] Visit `http://www.cs.ust.hk/~ivor/cvm.html`.

APPENDIX

D

Thesis Web Log

«The web is more a social creation than a technical one. I designed it for a social effect — to help people work together — and not as a technical toy. The ultimate goal of the Web is to support and improve our web-like existence in the world.»

TIM BERNERS-LEE
(BERNERS-LEE AND FISCHETTI, 1999, P. 123)

WORKING on a Master Degree thesis is not an easy job. One of the most hard task is maintaining clear and organized notes to keep track of our own work. Furthermore we have to distinguish and relate notes to each other, look for some notes, refer to previous notes, refer to some bibliography references and so on. This becomes increasingly more boring as the time elapse. A natural way to manage this aspect of the work is to keep a journal, a log. Neither digital nor papery documents are the best solution to adopt. They make hard the most frequently made operations, like editing, updating and searching; and we do not have date and time of annotations unless we write down them by hand. Finally, why not share this log with our own supervisors?

All of these questions led me to set up a Web Blog (or by using a portmanteau, a Blog) to keep track of my own work on this thesis and at the same time share such a log with my supervisors, bringing to the Web the natural social behavior that characterizes the relation between the student and his supervisors.

Figure D.1: *A screenshot of the Thesis Web Log.*

D.1 The Blog

The Blog was yet set up at early stages of the work. Its name is *Thesis Neminis* and it is located at `http://thesis.neminis.org`. The Blog is mostly in Italian language,[1] but it supports multi-language features and some of the content are yet translated in English (the most important ones, obviously). I expect to translate all the remaining relevant content in the future. All notes which will result irrelevant or erroneous at the end of the work will be marked with a tag "useless" or a similar one.

D.1.1 The contents

The core of the Blog is the notes, which are the essence of a log. There are also a variety of sections, like the information about the university and the professors, the documents produced during the work and the bibliography.

D.2 The features

A Blog provides many advantages to keep a high quality work-log. The main one rely on the Web basics: we can use the hypertext, which compares better with the classic text, because we can refer the annotations to each other or link

[1] It is my mother tongue and the one of my supervisors.

some external resources easily. In addition to the basic features provided by the hypertext, we have also other interesting features

- The supervisors (and other readers) can comment the notes directly.
- There is an integrated search engine which allows everyone to find fast what they are looking for.
- The content is "semantically" organized thanks to the tagging of the notes with a potentially unlimited number of keywords.
- The bibliography is automatically extracted (using a custom tag) from BibTeX files and presented in HTML format. For each entry we have the full citation, the link to the original publication (where available) and the link to the related BibTeX record.
- We can refer to a bibliography entry in a BibTeX file directly in notes, using a simple custom tag.
- The new notes are automatically notified to the supervisors.

D.3 The technology

The Blog is built on top of *Wordpress* platform,[2] a content management system specific for blogging. The automated management of the BibTeX files is provided by the Bib2HTML[3] Wordpress Plugin.

[2]For more information, visit `http://www.wordpress.org/`.
[3]Created by Sergio Andreozzi. Visit `http://sergioandreozzi.com/`.

BIBLIOGRAPHY

Aizerman, M. A., Braverman, E. M., and Rozonoer, L. I. (1964). The probability problem of pattern recognition learning and the method of potential functions. *Automation and Remote Control*, 25:1175–1190.

Ali, S. and Smith-Miles, K. A. (2006). A meta-learning approach to automatic kernel selection for Support Vector Machines. *Neurocomputing*, 70:173–186.

Ancona, N., Maglietta, R., and Stella, E. (2004). On sparsity of data representation in support vector machines. In *Signal and Image Processing*, volume 444, Via Amendola 122/D-I - 70126 Bari, Italy. Istituto di Studi sui Sistemi Intelligenti per l'Automazione - C.N.R.

Asharaf, S., Murty, M. N., and Shevade, S. K. (2007). Multiclass core vector machine. In *Proceedings of the 24th Annual International Conference on Machine Learning*.

Asharaf, S., Shevade, S. K., and Murty, N. M. (2006). Scalable rough support vector clustering. Technical report, Indian Institute of Science, Bangalore, 560012.

Asuncion, A. and Newman, D. (2007). UCI machine learning repository. [`http://www.ics.uci.edu/~mlearn/MLRepository.html`].

Bâdoiu, M. and Clarkson, K. L. (2003). Smaller core-sets for balls. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 801–802, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Baker, L. D. and McCallum, A. K. (1998). Distributional clustering of words for text classification. In Croft, W. B., Moffat, A., van Rijsbergen, C. J., Wilkinson, R., and Zobel, J., editors, *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 96–103, Melbourne, AU. ACM Press, New York, US.

Ban, T. and Abe, S. (2004). Spatially chunking support vector clustering algorithm. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 1.

Banerjee, A., Dhillon, I., Ghosh, J., Merugu, S., and Modha, D. (2004a). A generalized Maximum Entropy approach to Bregman co-clustering and matrix approximation. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD)*, pages 509–514.

Banerjee, A., Dhillon, I. S., Ghosh, J., Merugu, S., and Modha, D. (2004b). A generalized Maximum Entropy approach to Bregman co-clustering and matrix approximation. Technical report, UTCS TR04-24, UT, Austin.

Banerjee, A., Dhillon, I. S., Ghosh, J., Merugu, S., and Modha, D. S. (2007). A generalized Maximum Entropy approach to Bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8:1919–1986.

Banerjee, A., Guo, X., and Wang, H. (2005a). On the optimality of conditional expectation as a bregman predictor. *IEEE Transactions on Information Theory*, 51(7):2664–2669.

Banerjee, A., Krumpelman, C., Ghosh, J., Basu, S., and Mooney, R. J. (2005b). Model-based overlapping clustering. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 532–537, New York, NY, USA. ACM Press.

Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. (2004c). Clustering with Bregman Divergences. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, pages 234–245.

Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. (2005c). Clustering with Bregman Divergences. *Journal of Machine Learning Research*, 6:1705 – 1749.

Bellman, R. E. (1961). *Adaptive Control Processes: A Guided Tour*. Princeton University Press.

Ben-Hur, A., Ellisseeff, A., and Guyon, I. (2002). A stability based method for discovering structure in clustered data. In Altman, R., Dunker, A., Hunter, L., Klein, T., and Lauderdale, K., editors, *Pacific Symposium on Biocomputing*, volume 7, pages 6–17, Lihue, Hawaii, USA. World Scientific Pub. Co.

Ben-Hur, A., Horn, D., Siegelmann, H., and Vapnik, V. (2000a). A support vector method for hierarchical clustering. In *Fourteenth Annual Conference on Neural Information Processing Systems*, Denver, Colorado.

Ben-Hur, A., Horn, D., Siegelmann, H. T., and Vapnik, V. (2000b). A support vector method for clustering. In *Neural Information Processing Systems*, pages 367–373.

Ben-Hur, A., Horn, D., Siegelmann, H. T., and Vapnik, V. (2001). Support vector clustering. *Journal of Machine Learning Research*, 2:125–137.

Ben-Hur, A., Siegelmann, H. T., Horn, D., and Vapnik, V. (2000c). A support vector clustering method. *International Conference on Pattern Recognition*, 02:2724.

Berkhin, P. (2002). Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA.

Bertoni, A. and Valentini, G. (2006). Discovering significant structures in clustered data through bernstein inequality. In *Conferenza Italiana Sistemi Intelligenti (CISI 06)*, Ancona, Italy.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bolshakova, N. and Azuaje, F. (2005). Estimating the number of clusters in DNA microarray data. Technical report, Department of Computer Science, University of Dublin, Trinity College, Dublin, Ireland.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.

Bregman, L. M. (1967). The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217.

Brucker, P. (1978). On the complexity of clustering problems. *Lecture Notes in Economics and Mathematical Systems*, 157:45–54.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Minining and Knowledge Discovery*, 2(2):121–167.

Cai, R., Lu, L., and Cai, L.-H. (2005). Unsupervised auditory scene categorization via key audio effects and information-theoretic co-clustering. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2005*, volume 2, pages 1073–1076. ACL/SIGPARSE.

Camastra, F. (2004). *Kernel Methods for Unsupervised Learning*. PhD thesis, Università degli Studi di Genova, Dipartimento di Informatica e Scienze dell'Informazione.

Camastra, F. (2005). Kernel methods for clustering. In *Proceedings of 16th Workshop of Italian Neural Network Society (WIRN05)*, Lectures Notes on Computer Science Series, Vietri sul Mare, Italy. Springer-Verlag.

Camastra, F. (2007). A private e-mail conversation about the Kernel Grower clustering algorithm.

Camastra, F. and Verri, A. (2005). A novel kernel method for clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):801–805.

Carbonell, J. G., Michalski, R. S., and Mitchell, T. M. (1984). An overview of machine learning. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 3–23. Springer, Berlin, Heidelberg.

Chang, C.-C. and Lin, C.-J. (2007). Libsvm: A library for support vector machines. Manual available at `http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf`.

Cheng, Y. and Church, G. M. (2000). Biclustering of expression data. In *Intelligent Systems for Molecular Biology*, pages 93–103. AAAI Press.

Chiang, J.-C. and Wang, J.-S. (2004). A Validity-Guided Support Vector Clustering Algorithm for Identification of Optimal Cluster Configuration. *IEEE International Conference on Systems, Man and Cybernetics*, 4:3613–3618.

Chiang, J.-C. and Wang, J.-S. (2006). Support vector clustering with a novel cluster validity method. *Systems, Man and Cybernetics, 2006. ICSMC '06. IEEE International Conference on*, 5:3715–3720.

Chiang, J.-H. and Hao, P.-Y. (2003). A new kernel-based fuzzy clustering approach: support vector clustering with cell growing. *IEEE Transactions on Fuzzy Systems*, 11(4):518–527.

Cho, H., Dhillon, I., Guan, Y., and Sra, S. (2004a). Minimum sum squared residue co-clustering of gene expression data. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, pages 114–125.

Cho, H., Guan, Y., and Sra, S. (2004b). Co-cluster (v 1.1). Bregman co-clustering software.

Choo, J., Jiamthapthaksin, R., Chen, C., Celepcikay, O., Giusti, C., and Eick, C. (2007). Mosaic: A proximity graph approach to agglomerative clustering. In *9th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, Regensburg, Germany.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46.

Collobert, R., Bengio, S., and Bengio, Y. (2002). A parallel mixture of svms for very large scale problems. In *Advances in Neural Information Processing Systems*. MIT Press.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press, second edition.

Cover, T. M. and Thomas, J. A. (1991). *Elements of information theory*. Wiley-Interscience, New York, NY, USA.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines*. Cambridge University Press.

Csiszár, I. (1991). Why Least Squares and Maximum Entropy? An Axiomatic Approach to Inference for Linear Inverse Problems. *The Annals of Statistics*, 19(4):2032–2066.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227.

Della Pietra, S., Della Pietra, V., and Lafferty, J. (2001). Duality and auxiliary functions for Bregman distances. TR CMU-CS-01-109 TR CMU-CS-01-109, Carnegie Mellon University.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Deodhar, M., Cho, H., Gupta, G., Ghosh, J., and Dhillon, I. (2007a). Bregman Bubble Co-clustering. Technical report, Department of Electrical and Computer Engineering, The University of Texas at Austin.

Deodhar, M., Cho, H., Gupta, G., Ghosh, J., and Dhillon, I. S. (2007b). Bregman Bubble Co-clustering: A Framework for Detecting Dense Co-clusters in Large, High-Dimensional Data. *Submitted for publication*.

Deodhar, M. and Ghosh, J. (2007). A framework for simultaneous co-clustering and learning from complex data. Technical report, Intelligent Data Exploration & Analysis Laboratory, Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, Texas 78712.

Dhillon, I. and Guan, Y. (2003a). Information theoretic clustering of sparse co-occurrence data. In *Proceedings of The Third IEEE International Conference on Data Mining*, pages 517–520.

Dhillon, I., Guan, Y., and Kulis, B. (2005a). A fast kernel-based multilevel algorithm for graph clustering. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 629–634, New York, NY, USA. ACM Press.

Dhillon, I., Guan, Y., and Kulis, B. (2005b). A unified view of kernel k-means, spectral clustering and graph partitioning. Utcs technical report #tr-04-25, University of Texas at Austin, Department of Computer Science, UT, Austin.

Dhillon, I., Mallela, S., and Kumar, R. (2003a). A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 3:1265–1287.

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *Knowledge Discovery and Data Mining*, pages 269–274.

Dhillon, I. S., Fan, J., and Guan, Y. (2001). Efficient clustering of very large document collections. In R. Grossman, C. Kamath, V. K. and Namburu, R., editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers. Invited book chapter.

Dhillon, I. S. and Guan, Y. (2003b). Information theoretic clustering of sparse co-occurrence data. Technical report tr-03-39, The University of Texas at Austin, Department of Computer Sciences.

Dhillon, I. S., Guan, Y., and Kogan, J. (2002). Iterative clustering of high dimensional text data augmented by local search. *icdm*, 00:131.

Dhillon, I. S., Guan, Y., and Kulis, B. (2004). Kernel k-means, spectral clustering and normalized cuts. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 551–556.

Dhillon, I. S., Mallela, S., and Modha, D. S. (2003b). Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, pages 89–98.

Donalek, C. (2006). *Mining Astronomical Massive Data Sets*. PhD thesis, Università degli studi di Napoli "Federico II", Via Cinthia, 80126 Naples, Italy.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification*. Jhon Wiley & Sons, 2nd edition.

Dunn, J. C. (1974). Well separated clusters and optimal fuzzy partitions. *J. Cybernet*, 4:95–104.

Fan, R.-E., Chen, P.-H., and Lin, C.-J. (2005). Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188.

Forina, M., Leardi, R., Armanino, C., and Lanteri, S. (1988). PARVUS: An extendable package of programs for data exploration, classification and correlation. *Journal of Chemometrics*, 4(2):191–193.

Frawley, W. J., Piatetsky-Shapiro, G., and Matheus, C. J. (1992). Knowledge discovery in databases - an overview. *AI Magazine*, 13:57–70.

Freitag, D. (2004). Trained named entity recognition using distributional clusters. In *Conference on Empirical Methods on Natural Language Processing*, pages 262–269.

Garcia, C. and Moreno, J. A. (2004). Application of support vector clustering to the visualization of medical images. In *IEEE International Symposium on Biomedical Imaging: Nano to Macro*, volume 2, pages 1553–1556, Venezuela. Lab. de Computacion Emergente, Univ. Central de Venezuela.

Gennari, J. H., Langley, P., and Fisher, D. (1989). Models of incremental concept formation. *Artif. Intell.*, 40(1-3):11–61.

George, T. and Merugu, S. (2005). A Scalable Collaborative Filtering Framework Based on Co-Clustering. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 625–628, Washington, DC, USA. IEEE Computer Society.

Gokcay, E. and Principe, J. (2000). A new clustering evaluation function using Renyi's information potential. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3490–3493.

Goodman, L. and Kruskal, W. (1954). Measures of associations for cross-validations. *Journal of the American Statistical Association*, 64:732.

Gray, A. (1997). *Modern Differential Geometry of Curves and Surfaces with Mathematica (Second Edition)*. CRC.

Guan, J., Qiu, G., and Xue, X.-Y. (2005). Spectral images and features co-clustering with application to content-based image retrieval. In *IEEE 7th Workshop on Multimedia Signal Processing*, pages 1–4.

Guha, S., Rastogi, R., and Shim, K. (1998). CURE: an efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGKDD Conference*, pages 73–84, Seattle, WA.

Günter, S. and Bunke, H. (2003). Validation indices for graph clustering. *Pattern Recognition Letters*, 24(8):1107–1113.

Gupta, G. K. (2006). *Robust Methods for Locating Multiple Dense Regions in Complex Datasets*. PhD thesis, The University of Texas at Austin.

Gupta, R. and Chandola, V. (2006). Algorithms for Obtaining Overlapping Co-clusters. (CS8980) Machine Learning Project Mid-term Progress Report.

Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145.

Hansen, M. S., Sjöstrand, K., Ólafsdóttir, H., Larsson, H. B. W., Stegmann, M. B., and Larsen, R. (2007). Robust pseudo-hierarchical support vector clustering. In Ersbøll, B. K. and Pedersen, K. S., editors, *SCIA*, volume 4522 of *Lecture Notes in Computer Science*, pages 808–817. Springer.

Hao, P.-Y., Chiang, J.-H., and Tu, Y.-K. (2007). Hierarchically svm classification based on support vector clustering method and its application to document categorization. *Expert Syst. Appl.*, 33(3):627–635.

Hartigan, J. and Wong, M. (1979). Algorithm as136: A K-means clustering algorithm. *Applied Statistics*, 28:100–108.

Hartigan, J. A. (1972). Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129.

Hartono, H. and Widyanto, M. R. (2007). Angle Decrement Based Gaussian Kernel Width Generator for Support Vector Clustering. In *International Conference on Instrumentation, Communication and Information Technology*, Grand Aquila Hotel, Jl. Dr. Djundjunan 116, Bandung, West Java, Indonesia, 40173. Faculty of Computer Science, University of Indonesia.

Hettich, S. and Bay, S. D. (1999). The UCI KDD Archive. [`http://kdd.ics.uci.edu`].

Huang, J.-J., Tzeng, G.-H., and Ong, C.-S. (2007). Marketing segmentation using support vector clustering. *Expert Systems with Applications*, 32(2):313–317.

Hubert, L. J. and Schultz, J. V. (1976). Quadratic assignment as a general data analysis strategy. *British Journal of Mathematical and Statistical Psychology*, 29:190–241.

Huynh, T. and Vu, D. (2007). Using Co-clustering for Predicting Movie Rating in Netflix. Technical report, University of Texas at Austin, Department of Computer Sciences.

Jiang, D., Eick, C. F., and Chen, C. S. (2007). On supervised density estimation techniques and their application to spatial data mining. In *15th ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS)*, Seattle, Washington.

Joachims, T. (1998). Making large-scale support vector machine learning practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA.

Jong, K., Marchiori, E., and van der Vaart, A. (2003). Finding clusters using support vector classifiers. In *European Symposium on Artificial Neural Networks*, pages 223–228, The Netherlands. Department of Mathematics and Computer Science, Vrije Universiteit Amsterdam.

Kandogan, E. (2000). Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *Proceedings of the IEEE Information Visualization Symposium*, 650 Harry Road, San Jose, CA 95120. IBM Almaden Research Center.

Kandogan, E. (2001). Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 107–116, New York, NY, USA. ACM Press.

Karypis, G., Han, E.-H. S., and NEWS, V. K. (1999). CHAMELEON: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75.

Keerthi, S. S. and DeCoste, D. (2005). A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs. *J. Mach. Learn. Res.*, 6:341–361.

Kohonen, T. (1995). *Self-Organizing Maps*. Springer.

Kushmerick, N. (1999). Learning to remove internet advertisements. In *AGENTS '99: Proceedings of the third annual conference on Autonomous Agents*, pages 175–181, New York, NY, USA. ACM.

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

Lavielle, M. and Moulines, E. (1997). A simulated annealing version of the EM algorithm for non-gaussian deconvolution. *Statistics and Computing*, 7(4):229–236.

Lee, J. and Lee, D. (2005). An improved cluster labeling method for support vector clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):461–464. Member-Jaewook Lee.

Lee, J. and Lee, D. (2006). Dynamic characterization of cluster structures for robust and inductive support vector clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(11):1869–1874.

Lee, S.-H. and Daniels, K. M. (2004). Gaussian kernel width exploration in support vector clustering. Technical report, University of Massachussets Lowell, Department of Computer Science, Lowell, MA 01854.

Lee, S.-H. and Daniels, K. M. (2005a). Gaussian kernel width generator for support vector clustering. In He, M., Narasimhan, G., and Petoukhov, S., editors, *Advances in Bioinformatics and Its Applications*, volume 8, pages 151–162.

Lee, S.-H. and Daniels, K. M. (2005b). Gaussian kernel width selection and fast cluster labeling for support vector clustering. Technical report, Department of Computer Science, University of Massachussets Lowell.

Lee, S.-H. and Daniels, K. M. (2006). Cone cluster labeling for support vector clustering. In *Proceedings of 6th SIAM Conference on Data Mining*, pages 484–488.

Leisch, F., Weingessel, A., and Dimitriadou, E. (1998). Competitive learning for binary valued data. In Niklasson, L., Boden, M., and Ziemke, T., editors, *Proceedings of the 8th International Conference on Articial Neural Networks (ICANN 98)*, volume 2, pages 779–784, Skovde, Sweden.

Lekadir, K., Merrifield, R., and Yang, G. (2007). Outlier detection and handling for robust 3-D active shape models search. *IEEE Transaction On Medical Imaging*, 26(2):212–222.

Lin, H.-T. (2005). Infinite ensemble learning with support vector machines. Master's thesis, California Institute of Technology.

Linde, Y., Buzo, A., and Gray, R. M. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95.

Ling, P., Wang, Y., and Zhou, C. (2006). Self-adaptive two-phase support vector clustering for multi-relational data mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 225–229.

Ling, P. and Zhou, C. (2006). Adaptive support vector clustering for multi-relational data mining. In *Advances in Neural Networks - ISNN*, Lecture Notes in Computer Science, pages 1222–1230. Springer.

Long, B., Zhang, Z. M., and Yu, P. S. (2005). Co-clustering by block value decomposition. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 635–640, New York, NY, USA. ACM Press.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley. University of California Press.

Mangasarian, O. L., Setiono, R., and Wolberg, W. H. (1990). Pattern recognition via linear programming: theory and application to medical diagnosis. In Coleman, T. F. and Li, Y., editors, *Large-Scale Numerical Optimization*, pages 22–31, Philadelphia, Pennsylvenia. SIAM.

Manning, C. D., Raghavan, P., and Schütze, H. (2007). *Introduction to Information Retrieval*. Cambridge University Press.

Martinetz, T. and Schulten, K. (1994). Topology representing networks. *Neural Netw.*, 7(3):507–522.

Martinetz, T. M. and Schulten, K. J. (1991). A neural-gas network learns topologies. In Kohonen, T., Mäkisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, pages 397–402, North-Holland.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bul letin of Mathemathical Biophysics*, 5:115–133.

Merz, P. (2003). An iterated local search approach for minimum sum-of-squares clustering. In *Advances in Intelligent Data Analysis V (IDA 2003)*, volume 2810 of *Lecture Notes in Computer Science*, pages 286–296, Heidelberg. Springer.

Minsky, M. and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. The MIT Press.

Modha, D. S. and Spangler, W. S. (2003). Feature weighting in k-means clustering. *Machine Learning*, 52(3):217–237.

Nasser, A., Hamad, D., and Nasr, C. (2006). K-means clustering algorithm in projected spaces. In *9th International Conference on Information Fusion*, pages 1–6, Florence, Italy.

Nath, J. S. and Shevade, S. K. (2006). An efficient clustering scheme using support vector methods. *Pattern Recognition*, 39(8):1473–1480.

Nielsen, F. and Nock, R. (2006). On approximating the smallest enclosing Bregman Balls. In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 485–486, New York, NY, USA. ACM Press.

Nilsson, N. J. (2005). Introduction to machine learning. *Draft of Incomplete Notes*.

Nock, R. and Nielsen, F. (2005). Fitting the smallest enclosing Bregman balls. In *16th European Conference on Machine Learning*, number 3720 in Lectures Notes on Computer Science Series, pages 649–656. Springer-Verlag.

Orlandic, R., Lai, Y., and Yee, W. G. (2005). Clustering high-dimensional data using an efficient and effective data space reduction. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 201–208, New York, NY, USA. ACM.

Park, J., Ji, X., Zha, H., and Kasturi, R. (2004). Support vector clustering combined with spectral graph partitioning. In *Proceedings of the 17th International Pattern Recognition*, volume 4, pages 581–584. Departement of Computer Science and Engineering, Pennsylvania State University, PA, USA.

Parsons, L., Haque, E., and Liu, H. (2004). Subspace clustering for high dimensional data: A review. *SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining*, 6(1):90.

Pei, Y. and Zaïane, O. (2006). A synthetic data generator for clustering and outlier analysis. Technical report, Computing Science Department, University of Alberta, Edmonton, Canada T6G 2E8.

Pelleg, D. and Moore, A. (2000). X-means: Extending K-means with efficient estimation of the number of clusters. In *In Proceedings of 17th International Conference on Machine Learning*, pages 727–734. Morgan Kaufmann, San Francisco, CA.

Pereira, F. C. N., Tishby, N., and Lee, L. (1993). Distributional clustering of english words. In *Meeting of the Association for Computational Linguistics*, pages 183–190.

Ping, L., Yan, W., Nan, L., Yu, W. J., Shuang, L., and ChunGuang, Z. (2005). Two-phase support vector clustering for multi-relational data mining. In *International Conference on Cyberworlds*, pages 139–146, Washington, DC, USA. Coll. of Comput. Sci., Jilin Univ., Changchun, China, IEEE Computer Society.

Platt, J. C. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, USA.

Puma-Villanueva, W. J., Bezerra, G. B., Lima, C. A. M., and Zuben, F. J. V. (2005). Improving support vector clustering with ensembles. In *Internation Joint Conference on Neural Networks*, pages 13–15, Montreal, Quebec, CANADA.

Qiu, G. (2004). Image and feature co-clustering. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4*, pages 991–994, Washington, DC, USA. IEEE Computer Society.

Rennie, J. D. M. (2001). Improving multi-class text classification with naive bayes. Master's thesis, Massachusetts Institute of Technology.

Ribeiro, B. (2002). On data based learning using support vector clustering. In *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02.*, volume 5, pages 2516–2521.

Roberts, S. J. (1996). Parametric and non-parametric unsupervised cluster analysis. *Pattern Recognition*, 30(2):261–272.

Rohwer, R. and Freitag, D. (2004). Towards full automation of lexicon construction. In *In Proceedings of HLT-NAACL 04: Computational Lexical Semantics Workshop*, pages 9–16, Boston, MA.

Rosenblatt, F. (1963). Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. *The American Journal of Psychology*, 76(4):705–707.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. *Paralle Distribute Processing*, pages 318–362.

Sammon, J. W., J. (1969). A nonlinear mapping for data structure analysis. *Transactions on Computers*, C-18(5):401–409.

Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., and Williamson, R. (1999). Estimating the support of a high-dimensional distribution. Technical Report 99–87, Microsoft Research, Redmond, WA.

Schölkopf, B., Smola, A., Williamson, R. C., and Bartlett, P. L. (2000a). New support vector algorithms. *Neural Computation*, 12:1083–1121.

Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., and Platt, J. (2000b). Support vector method for novelty detection. In *Advances in Neural Information Processing Systems 12: Proceedings of the 1999 Conference*.

Shafiei, M. and Milios, E. (2006). Model-based Overlapping Co-Clustering. In *Proceedings of the Fourth Workshop on Text Mining, Sixth SIAM International Conference on Data Mining*, Bethesda, Maryland.

Slonim, N. and Tishby, N. (2000). Document clustering using word clusters via the information bottleneck method. In *Research and Development in Information Retrieval*, pages 208–215.

Slonim, N. and Tishby, N. (2001). The power of word clusters for text classification. In *23rd European Colloquium on Information Retrieval Research*.

Smola, A. J., Schölkopf, B., and Müller, K.-R. (1998). The connection between regularization operators and support vector kernels. *Neural Networks*, 11(4):637–649.

Suykens, J. A. K., Gestel, T. V., Brabanter, J. D., and B. De Moor, J. V. (2002). *Least Squares Support Vector Machines*. World Scientific Pub. Co., Singapore.

Tax, D. M. J. (2001). *One-class classification: concept learning in the absence of counter-examples*. PhD thesis, Technische Universiteit Delft.

Tax, D. M. J. (2007). Ddtools, the data description toolbox for matlab. Version 1.6.1.

Tax, D. M. J. and Duin, R. P. W. (1999a). Data domain description using support vectors. In *European Symposium on Artificial Neural Network*, pages 251–256, Bruges (Belgium).

Tax, D. M. J. and Duin, R. P. W. (1999b). Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199.

Tax, D. M. J. and Duin, R. P. W. (2004). Support vector data description. *Machine learning*, 54(1):45–66.

Tax, D. M. J. and Juszczak, P. (2003). Kernel whitening for one-class classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(3):333–347.

Tchagang, A. B. and Tewfik, A. H. (2005). Robust biclustering algorithm (ROBA) for DNA microarray data analysis. In *13th IEEE Workshop on Statistical Signal Processing*, pages 984–989.

Teboulle, M., Berkhin, P., Dhillon, I. S., Guan, Y., and Kogan, J. (2005). Clustering with Entropy-like k-means Algorithms. In Kogan, J., Nicholas, C., and Taboulle, M., editors, *Grouping Multidimensional Data: Recent Advances in Clustering*, chapter 5, pages 127–160. Springer-Verlag.

Tsang, I. W., Kocsor, A., and Kwok, J. T. (2007). Simpler core vector machines with enclosing balls. In *Twenty-Fourth International Conference on Machine Learning (ICML)*, Corvallis, Oregon, USA.

Tsang, I. W., Kwok, J. T., and Cheung, P.-M. (2005). Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392.

Tsang, I. W., Kwok, J. T., and Zurada, J. M. (2006). Generalized core vector machines. *IEEE Transactions on Neural Networks*, 17(5):1126–1140.

Tung, J.-W. and Hsu, C.-T. (2005). Learning hidden semantic cues using support vector clustering. In *IEEE Internation Conference on Image Processing*, volume 1, pages 1189–1192.

Vapnik, V. N. (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

Vapnik, V. N. (2000). *The Nature of Statistical Learning Theory*. Springer, 2nd edition.

Various (2007). Boost C++ Libraries. Visit `http://www.boost.org`.

Vennam, J. R. and Vadapalli, S. (2005). Syndeca: A tool to generate synthetic datasets for evaluation of clustering algorithms. In *11th International Conference on Management of Data (COMAD 2005)*, Goa, India. http://cde.iiit.ac.in/syndeca.

Wagstaff, K. (2004). Clustering with missing values: No imputation required. In *Classification, Clustering, and Data Mining Applications (Proceedings of the Meeting of the International Federation of Classification Societies)*, pages 649–658, 4800 Oak Grove Dr., Pasadena, CA 91109. Jet Propulsion Laboratory, California Institute of Technology.

Wagstaff, K. and Laidler, V. G. (2005). Making the most of missing values: Object clustering with partial data in astronomy. In Shopbel, P. L., Britton, M. C., and Ebert, R., editors, *Astronomical Data Analysis Software and Systems XIV (ADASS)*, volume 347 of *ASP Conference Series*.

Wang, D., Shi, L., Yeung, D. S., Heng, P.-A., Wong, T.-T., and Tsang, E. C. C. (2005). Support vector clustering for brain activation detection. In *Proceedings of the 8th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI 2005)*, pages 572–579, Palm Springs, California, USA.

Wang, J.-S. and Chiang, J.-C. (2008). A cluster validity measure with a hybrid parameter search method for the support vector clustering algorithm. *Pattern Recognition*, 41(2):506–520.

Waters, A. and Raghavan, A. (2007). Missing Value Estimation with Bregman Co-clustering for Sound Localization. Final Report on Inter-aural intensity difference imputation with co-clustering.

Welzl, E. (1991). Smallest enclosing disks (balls and ellipsoids). In Maurer, H., editor, *New Results and New Trends in Computer Science*, LNCS. Springer.

Westin, L. K. (2004). Missing data and the preprocessing perceptron. Technical report, Department of Computing, UmeåUniversity, SE-90187 Umeå, Sweden.

Yang, J., Estivill-Castro, V., and Chalup, S. K. (2002). Support vector clustering through proximity graph modelling. In *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02.*, volume 2, pages 898–903.

Yang, J., Wang, H., Wang, W., and Yu, P. (2003). Enhanced biclustering on expression data. In *Proceedings of the 3rd IEEE Conference on Bioinformatics and Bioengineering (BIBE)*, pages 321–327.

Yankov, D., Keogh, E., and Kan, K. (2007). Locally constrained support vector clustering. In *IEEE Internation Conference on Data Mining*.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8:338–353.

Zaïane, O. R., Foss, A., Lee, C.-H., and Wang, W. (2002). On data clustering analysis: Scalability, constraints, and validation. In *PAKDD '02: Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 28–39, London, UK. Springer-Verlag.

Zheng, E.-H., Yang, M., Li, P., and Song, Z.-H. (2006). Fuzzy support vector clustering. In *Advances in Neural Networks - ISNN*, Lecture Notes in Computer Science, pages 1050–1056. Springer.

The bibliography of this thesis is also available online at `http://thesis.neminis.org/bibliografia/`, with links to the original publications (where abailable) and the related BibTeX entries.

EPIGRAPHS REFERENCES

Berners-Lee, T. and Fischetti, M. (1999). *Weaving the Web : The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper San Francisco.

Dijkstra, E. W. (1982). *Selected writings on computing: a personal perspective*. Springer-Verlag New York, Inc., New York, NY, USA.

Fisher, R. A. (1956). *Statistical Methods and Scientific Inference*. Hafner Pub. Co.

Knuth, D. E. (1968). *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Reading: Addison Wesley Publishing Co.

Knuth, D. E. (1974). Computer programming as an art. *Communications of the ACM*, 17(12):667–673.

Laplace, P. S. (1812). *Théorie analytique des probabilités*. M.me V.e Courcier, imprimeur-libraire pour les mathématiques, quai de Augustins, n. 57, 1st edition.

MacHale, D. (1993). *Comic Sections: Book of Mathematical Jokes, Humour, Wit and Wisdom*. Boole Press Ltd.

McIrvine, E. C. and Tribus, M. (1971). Energy and information (thermodynamics and information theory). *Scientific American*, 225(3):179–188.

Naso, P. O. (10). *Epistulae Ex Ponto*.

Spencer, H. (1862). *First Principles*.

Turing, A. (1939). System of logic based on ordinals. In *Proceedings of the London Mathematical Society*, volume 45, pages 161–228.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.