Cartographie de l'espace chimique

Héléna Alexandra Gaspar

UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE DES SCIENCES CHIMIQUES

Chimie de la matière complexe – UMR 7140

THÈSE présentée par :

Héléna Alexandra GASPAR

soutenue le : **29 septembre 2015**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline / Spécialité : **Chimie / Chémoinformatique**

Cartographie de l'espace chimique

THÈSE dirigée par :

Prof. VARNEK Alexandre Directeur de recherche, université de Strasbourg

RAPPORTEURS :

Prof. TABOUREAU Olivier Directeur de recherche, université Paris Diderot

Dr. ERTL Peter Docteur, Instituts Novartis pour la recherche biomédicale

---

AUTRES MEMBRES DU JURY :

Prof. ROGNAN Didier Directeur de recherche, université de Strasbourg

UNIVERSITÉ DE STRASBOURG

Héléna Alexandra Gaspar

Cartographie de l'espace chimique

EDSC École Doctorale des Sciences Chimiques

Résumé

Cette thèse est consacrée à la cartographie de l'espace chimique ; son but est d'établir les bases d'un outil donnant une vision d'ensemble d'un jeu de données, comprenant prédiction d'activité, visualisation, et comparaison de grandes librairies. Dans cet ouvrage, nous introduisons des modèles prédictifs QSAR (relations quantitatives structure à activité) avec de nouvelles définitions de domaines d'applicabilité, basés sur la méthode GTM (generative topographic mapping), introduite par C. Bishop et al. Une partie de cette thèse concerne l'étude de grandes librairies de composés chimiques grâce à la méthode GTM incrémentale. Nous introduisons également une nouvelle méthode « Stargate GTM », ou S-GTM, permettant de passer de l'espace des descripteurs chimiques à celui des activités et *vice versa*, appliquée à la prédiction de profils d'activité ou aux QSAR inverses.

Mots-clés : visualisation, espace chimique, QSAR, inverse QSAR, domaine d'applicabilité, GTM, données massives

Résumé en anglais

This thesis is dedicated to the cartography of chemical space; our goal is to establish the foundations of a tool offering a complete overview of a chemical dataset, including visualization, activity prediction, and comparison of very large datasets. We introduce new QSAR models (quantitative structure-activity relationship) based on the GTM method (generative topographic mapping), introduced by C. Bishop et al. A part of this thesis is dedicated to the visualization and analysis of large chemical libraries using the incremental version of GTM. We also introduce a new method coined "Stargate GTM" or S-GTM, which allows us to travel from the space of chemical descriptors to activity space and vice versa; this approach was applied to activity profile prediction and inverse QSAR.

Key words: visualization, chemical space, QSAR, inverse QSAR, applicability domain, GTM, big data

UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE DES SCIENCES CHIMIQUES

Chimie de la matière complexe – UMR 7140

THÈSE présentée par :

Héléna Alexandra GASPAR

soutenue le : **29 septembre 2015**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline / Spécialité : **Chimie / Chémoinformatique**

Cartographie de l'espace chimique

THÈSE dirigée par :

Prof. VARNEK Alexandre	Directeur de recherche, université de Strasbourg

RAPPORTEURS :

Prof. TABOUREAU Olivier	Directeur de recherche, université Paris Diderot
Dr. ERTL Peter	Docteur, Instituts Novartis pour la recherche biomédicale

---

AUTRES MEMBRES DU JURY :

Prof. ROGNAN Didier	Directeur de recherche, université de Strasbourg

Cartography of Chemical Space

Héléna Alexandra Gaspar

Laboratoire de Chémoinformatique

A thesis submitted on September, 2015

for the Degree of Docteur de l'université de Strasbourg

Abstract

This thesis is dedicated to the visualization of chemical space; its purpose is to lay the foundations for a multitask toolbox providing an overview of a chemical dataset, including activity prediction, visualization, and comparison of large chemical libraries. We based most of our methods on the generative topographic mapping approach or GTM, introduced by C. Bishop et al. and mainly used until now for visualization and clustering tasks. In this thesis, we apply for the first time GTM as a regression and classification method, and introduce several applicability domain definitions for predictive models; we also discuss new methodologies for analyzing and comparing large chemical libraries, and show how to use GTM maps to perform inverse QSAR (quantitative structure-activity relationship) tasks. We also introduce a new method coined "Stargate GTM" or S-GTM, employed to predict activity profiles from molecular descriptors (classical QSAR), or molecular descriptors from activity profiles (inverse QSAR). GTM variants together with our own methodologies and algorithms were implemented in the software ISIDA/GTM developed for this thesis.

Declaration

- This thesis is my own work.
- Information sources were acknowledged and fully referenced.

Héléna Alexandra Gaspar
September, 2015

Acknowledgements

I thank my supervisor, Prof. A. Varnek, for giving me the opportunity to work on such a fascinating subject, as well as Prof I. I. Baskin, Dr. G. Marcou and Dr. H. Horvath, for their numerous and precious advises. I also thank everyone from the lab: Olga, thank you so much for your good mood and your constant care for others; Ioana, Tanya, Fanny, Fiorella, thanks for being such precious and generous friends; I also thank Christophe and Laurent who gracefully shared their office with me for a year - I only keep good memories from that time, and Grace, for all our conversations; thank you also Natalia, Aurélie, Julien, Pavel, Timur, Kirill, Birgit, my former interns Florian and Adrien, and there are surely others I forgot to mention. To my family in Paris, Serbia and Guadeloupe, my brothers Karl and Wilfried, my sister Emma, my wonderful parents Alexandre-Pierre and Béatrice, my dear grandmothers Milanka and Nadia, and my beloved late grandfathers Pierre and Joseph; to Gabriel who brought back some music into my life; to my faithful friends Arnaud, Hombeline, my music partner Gabriella, people from the EVUS university choir, and all my friends from Paris who did not see a lot of me these past 3 years: YOU ARE THE ROOTS OF MY LEARNING TREE, THANK YOU ALL FOR BEING YOU.

Contents

Summary in French **19**

1 Introduction **29**

2 Chemical Space and Chemical Universe **31**

3 Visualize a Multidimensional Space **35**
3.1 Descriptors 35
3.2 Apparent and Intrinsic Dimensionality 36
3.3 The Curse of Dimensionality 37
3.4 Dimensionality Reduction 39
3.4.1 Linear Dimensionality Reduction 39
3.4.2 Non-Linear Dimensionality Reduction 44
3.4.3 Visual Separation of Classes 53
3.5 Generative Topographic Mapping 55
3.5.1 Original GTM Algorithm 55
3.5.2 Kernel GTM 57
3.5.3 Incremental GTM 58
3.6 InfoVis Techniques for High-Dimensional Data 60
3.6.1 Introduction 60
3.6.2 Some InfoVis Examples 61
3.6.3 Conclusion 65

4 Big Data Problem **67**
4.1 Technical Considerations 67
4.2 Initialize iGTM with a Small Subset 68
4.2.1 Introduction 68
4.2.2 Methods 68
4.2.3 Results and Discussion 69
4.2.4 Conclusion 73

5 Visualizing Descriptors or Properties **75**
5.1 Activity or Descriptor Landscapes . . . 75
5.2 Descriptor Scores . . . 76
5.3 Regions of Interest (ROIs) . . . 79

6 Optimization of models **81**
6.1 Training and Test . . . 81
6.2 Model Validation Procedure . . . 82
6.3 GTM Parameters . . . 83
6.4 GTM Unsupervised Optimization . . . 85
6.5 GTM Supervized Optimization . . . 86

7 GTM Classification Models **89**
7.1 Initial Space Classification . . . 89
7.2 Latent Space Classification . . . 90

8 GTM Regression Models **93**
8.1 Unsupervised GTM Regression . . . 93
8.2 Supervised GTM Regression with S-GTM . . . 94

9 GTM Applicability Domain **95**
9.1 Applicability Domain and Outliers . . . 95
9.2 Structure-Dependent AD . . . 97
9.2.1 Likelihood-Based . . . 97
9.2.2 Density-Based . . . 97
9.3 Activity-Dependent AD . . . 98
9.4 Classification . . . 98
9.5 Regression . . . 100

10 Chemical Libraries Analysis **103**
10.1 Cumulated Responsibilities . . . 103
10.2 Library Diversity . . . 104
10.3 Comparing Landscapes . . . 104
10.4 Library Similarity . . . 104
10.5 Meta-GTM . . . 105

11 Stargate GTM (S-GTM) **107**
11.1 Introduction . . . 107
11.2 Stargate GTM . . . 108
11.2.1 Algorithm . . . 108

11.2.2 Prediction with S-GTM . . . 111
11.3 Application 1: Prediction of 8 Affinities . . . 111
11.3.1 Data and Descriptors . . . 111
11.3.2 Methods . . . 112
11.3.3 Results and Discussion . . . 113
11.3.4 Conclusion . . . 118
11.4 Application 2: Prediction of MOE 2D Descriptors . . . 118
11.4.1 Data and Descriptors . . . 118
11.4.2 Methods . . . 119
11.4.3 Results and Discussion . . . 119
11.4.4 Conclusion . . . 120
11.5 Going Further: Impact of Weight Factors . . . 120
11.6 Going Further: Impact of Property Correlation . . . 122
11.7 Conclusion . . . 125

12 Inverse QSAR **127**
12.1 Introduction . . . 127
12.2 Data and Descriptors . . . 127
12.3 Inverse QSAR with Conventional GTM . . . 128
12.3.1 Latent Prototype . . . 129
12.3.2 Manifold Prototype . . . 133
12.3.3 Landscape Prototype . . . 134
12.3.4 Conclusion . . . 135
12.4 Inverse QSAR with S-GTM . . . 135
12.4.1 Multimodality Problem . . . 136
12.4.2 Multiple Prototypes . . . 137
12.4.3 Discarded Method: Weighted Landscape Prototype . . . 138
12.4.4 Conditional Log Likelihood Approach . . . 139
12.5 Comparing GTM and S-GTM approaches . . . 141
12.6 Conclusion . . . 142

13 ISIDA/GTM Software **143**
13.1 Introduction . . . 144
13.2 Data and Set-Up . . . 145
13.3 Descriptors . . . 146
13.4 Build New GTM . . . 147
13.5 QSAR . . . 148
13.6 Visualization . . . 149

13.7 Use Case 1 - Ache Dataset 151
13.8 Use Case 2 - Gd^{3+} Dataset 152
13.9 Conclusion 154

14 Application of GTM Initial Space Classification to DUD **155**

15 Application of GTM Latent Space Classification to BDDCS **157**

16 Application of GTM Regression to QSAR **159**

17 Application of iGTM to 37 Chemical Libraries **161**

18 Conclusion **163**

Bibliography **165**

List of Figures **175**

List of Tables **178**

Glossary **180**

Symbols (GTM and variants) **191**

Appendix A Visualization of the SILIRID Space **193**

Appendix B GTMapTool 2015 Software Options **194**

Ἁρμονίη ἀφανὴς φανερῆς κρείττων.

The invisible harmony is more powerful then the visible.

Heraclitus.

Résumé en français

Introduction

Cette thèse est consacrée à la cartographie de l'espace chimique ; son but est d'établir les bases d'un outil donnant une vision d'ensemble d'un jeu de données, comprenant prédiction d'activité, visualisation, et comparaison de grandes librairies. Nous nous sommes basés sur la méthode GTM (generative topographic mapping) pour développer la plupart de nos approches méthodologiques. Cette méthode, développée par C. Bishop, a été jusqu'à présent utilisée pour des tâches de visualisation et de partitionnement des données. Dans cette thèse, nous démontrons pour la première fois l'usage de la cartographie GTM comme méthode de modélisation structure-activité (QSAR), et définissons plusieurs types de domaines d'applicabilité pour nos modèles ; nous l'appliquons également pour la première fois à des tâches de QSAR inverses. Une partie consacrée aux grands jeux de données concerne la comparaison de grandes librairies de composés chimiques grâce à la méthode GTM incrémentale. Nous introduisons également une toute nouvelle méthode « Stargate GTM », ou S-GTM, permettant de passer de l'espace des descripteurs chimiques à celui des activités et *vice versa*, utilisée d'une part pour la prédiction de profils d'activités à partir de structures chimiques, d'autre part pour rechercher des structures correspondant à un profil d'activité.

Définition d'un espace chimique à l'aide de descripteurs

En chémoinformatique, un espace chimique peut être considéré comme un ensemble de molécules décrites par des descripteurs moléculaires. Ces descripteurs, typiquement des caractéristiques structurales ou physico-chimiques des molécules, peuvent avoir des valeurs binaires, continues ou discrètes, et leur nombre peut atteindre plusieurs milliers. On imagine alors les composés chimiques comme des points dans un espace de dimensionnalité D égale au nombre de descripteurs ; D étant généralement largement supérieure à 3, il est impossible de visualiser un tel espace. C'est ici qu'interviennent les méthodes de réduction de dimension, permettant de réduire le nombre de dimensions à 2 ou 3, et de créer des cartes facilement interprétables. En chémoinformatique, les méth-

odes les plus couramment utilisées sont l'ACP (analyse en composantes principales), les cartes de Kohonen, et depuis peu la cartographie GTM. L'ACP et la cartographie GTM permettent de visualiser les composés chimiques comme autant de points sur une carte en 2 dimensions. Le grand avantage de l'approche GTM est son fondement probabiliste, permettant non seulement la visualisation, mais également la prédiction et la comparaison de jeux de données, une seule molécule étant associée à une distribution de probabilité sur la carte entière.

La Cartographie GTM et ses variantes

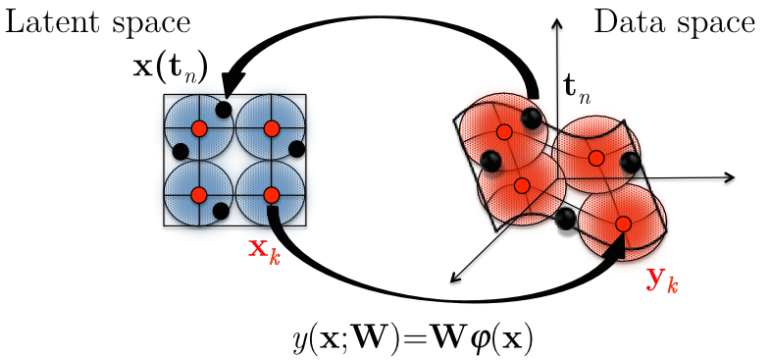


Figure 1: La méthode GTM, à l'aide de la fonction de projection $y(\mathbf{x};\mathbf{W})$, projette les points d'une grille en 2D $\mathbf{X}$ dans l'espace des données ($\mathbf{x}_k \rightarrow \mathbf{y}_k$) ; les distances entre les points $\{\mathbf{y}_k\}$ et les instances dans l'espace des données $\{\mathbf{t}_n\}$ permettent alors de calculer les coordonnées des instances $\{\mathbf{x}(\mathbf{t}_n)\}$ dans l'espace latent ($\mathbf{t}_n \rightarrow \mathbf{x}(\mathbf{t}_n)$).

La méthode GTM, introduite par C. Bishop, permet de passer de l'espace en D dimensions à un espace latent en 2 dimensions. L'espace en 2 dimensions est discrétisé en un ensemble de points $\{\mathbf{x}_k\}$ régulièrement placés sur une grille qu'on appellera « nœuds ». Le modèle génère un ensemble de responsabilités $\{R_{kn}\}$ ou probabilités de trouver une molécule $\mathbf{t}_n$ près d'un nœud $\mathbf{x}_k$ sur la carte en 2 dimensions. Le modèle est optimisé par un algorithme espérance-maximisation, optimisant une fonction de vraisemblance. Plusieurs variantes de la méthode GTM ont été décrites par d'autres auteurs, notamment « kernel GTM » (KGTM) et « incremental GTM » (iGTM) ; cette dernière est conçue pour pouvoir traiter des millions de composés de manière incrémentale. Tous ces algorithmes ont été implémentés dans l'outil en ligne de commande GTMapTool développé au cours de la thèse.

Visualisation et analyse de données

Nous proposons quelques approches originales de visualisation et d'analyse de données, parmi lesquelles les paysages d'activité et les régions d'intérêt. Comme les cartes géo-

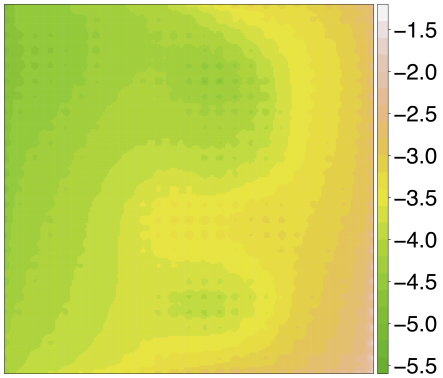


Figure 2: Exemple de paysage d'activité, représentant les valeurs de la solubilité (LogS) sur une carte GTM construite à partir d'une base de données d'environ 2,2 millions de molécules.

graphiques, une carte provenant d'un algorithme de réduction de dimension (GTM, ACP, Kohonen, MDS, etc.) peut être coloriée en fonction de l'activité des composés, de leurs classes ou même de leurs descripteurs. On obtient alors des paysages d'activité permettant d'identifier en un coup d'œil des régions particulières de l'espace chimique. L'approche GTM, générant des responsabilités $\{R_{kn}\}$ associées à chaque molécule $\mathbf{t}_n$ et nœud $\mathbf{x}_k$ sur la carte, permet de construire simplement ces paysages d'activité en calculant une valeur moyenne d'activité $\bar{a}_k$ à chaque nœud $\mathbf{x}_k$. Ces paysages peuvent être superposés pour trouver des régions autour des nœuds $\{\mathbf{x}_k\}$ correspondant à plusieurs critères à la fois, et relever les composés qui s'y trouvent.

Modèles de classification GTM

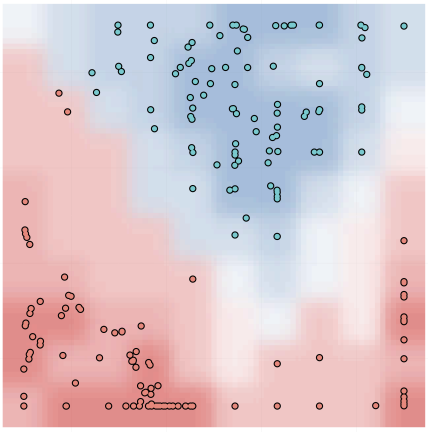


Figure 3: Exemple de modèle de classification GTM, visualisable sur une carte, où les couleurs bleues et rouges représentent deux classes (inhibiteurs ou non de l'acétylcholinestérase), et la transparence la confiance accordée à la prédiction.

La cartographie GTM permet de construire des modèles de classification dans l'espace

initial et latent. Dans notre article [1] paru dans Molecular Informatics en 2012, nous avons montré comment construire ces modèles dans l'espace initial, en utilisant des jeux de données provenant du Directory of Useful Decoys. Durant cette thèse, nous avons également établi deux méthodologies de classification par GTM dans l'espace 2D : la méthode du nœud le plus proche, et une méthode bayésienne globale. La méthode du nœud le plus proche consiste à calculer la probabilité conditionnelle $P(\mathbf{x}_k|c_i)$ de chaque nœud $\mathbf{x}_k$ sachant la classe c_i, à partir des responsabilités des composés. Les nœuds $\{\mathbf{x}_k\}$ sont alors coloriés en fonction de la classe la plus probable, et la classe prédite pour un nouveau composé $\mathbf{t}_q$ projeté sur la carte sera celle du nœud le plus proche. Le modèle bayésien global utilise les responsabilités pour calculer une probabilité conditionnelle $P(c_i|\mathbf{x}_k)$ de chaque classe c_i sachant la position $\mathbf{x}_k$ sur la carte ; la classe d'un nouveau composé $\mathbf{t}_q$ sera celle qui aura la plus grande probabilité $P(c_i|\mathbf{t}_q)$ obtenue à partir des probabilités $P(c_i|\mathbf{x})$ et des responsabilités de $\mathbf{t}_q$. Ces méthodes de classification dans l'espace latent ont été utilisées pour visualiser et analyser le BDDCS (Biopharmaceutical Drug Disposition Classification System), ou système de classification selon la solubilité et le métabolisme, dans notre article [2] paru dans le *Journal of Chemical Information and Modeling* en 2013 ; la performance des modèles obtenus était proche de celle de méthodes d'apprentissage automatique classiques.

Modèles de régression GTM

Plusieurs méthodes de régression ont été développées : une méthode de V plus proches voisins, et deux méthodes basées sur le paysage d'activité, l'une globale, l'autre locale. Les méthodes basées sur le paysage d'activité passent par le calcul des activités $\{\bar{a}_k\}$ attribuées à chaque nœud sur la carte. Pour la méthode locale, l'activité prédite d'une molécule $\mathbf{t}_q$ est la moyenne des activités des V plus proches nœuds dans l'espace à deux dimensions. L'activité prédite par la méthode globale est la moyenne des activités du paysage pondérées par les responsabilités de la molécule $\mathbf{t}_q$. Ces méthodes ont été comparées dans notre article [3] paru dans *Molecular Informatics* en 2015, utilisant différents jeux de données : inhibiteurs de la thrombine, solubilité, et complexants des métaux Gd^{3+}, Lu^{3+}, et Ca^{2+} ; les performances se sont révélées proches des méthodes d'apprentissage automatique classiques.

Domaine d'applicabilité de modèles GTM

Le domaine d'applicabilité (DA) permet d'estimer dans quelle mesure on peut appliquer un modèle QSAR. Les méthodes de réduction de dimension telles que la cartographie GTM permettent de visualiser les zones en dehors ou à l'intérieur du DA. Plusieurs techniques ont été explorées pour trouver les molécules en dehors du DA, comprenant

les molécules avec une faible vraisemblance, les molécules se trouvant dans une zone peu peuplée par l'ensemble d'entraînement, et enfin celles se trouvant dans les zones où la prédiction de classe ou d'activité est peu sûre. Des applications de domaine d'applicabilité sont données dans nos deux articles [3] et [2], l'un sur les modèles QSAR et l'autre sur la classification BDDCS. L'établissement d'un DA selon la vraisemblance existait déjà ; dans cette thèse, nous avons introduit les DA spécifiques à la classification et à la régression GTM.

Analyse de grands jeux de données par iGTM

L'iGTM (incremental GTM) peut être utilisé pour cartographier des millions de composés à la fois. La méthode GTM générant des responsabilités (probabilités a posteriori) pour toutes les molécules sur une carte en 2 dimensions, il est possible de résumer toute une librairie chimique par un simple vecteur de responsabilités, en cumulant les responsabilités de tous les composés qui la constituent. Ces vecteurs peuvent ensuite être comparés avec une simple mesure de similarité, ou être réutilisés pour créer de nouvelles cartes à un niveau supérieur d'abstraction, des « μGTMs » ou meta-GTMs, sur lesquelles un point ne sera non plus un composé, mais une librairie. Ces vecteurs-librairies peuvent aussi être utilisés pour évaluer l'entropie ou le niveau de dispersion d'une librairie dans l'espace en deux dimensions. Dans notre article [4] paru en 2015 dans le *Journal of Chemical Information and Modeling*, nous avons appliqué ces méthodes à la cartographie et à l'analyse de 37 librairies chimiques, totalisant un ensemble d'environ 2,2 millions de molécules. L'iGTM avait déjà été décrit par C. Bishop ; dans cette thèse, nous l'appliquons aux bases de données chimiques, et introduisons le concept de vecteurs-prototypes pour des bases de données et leur utilisation pour calculer la similarité, ainsi que le concept de μGTM.

Stargate GTM et profilage d'activité

Dans cet ouvrage, nous introduisons pour la première fois la méthode S-GTM. Partie d'une idée originale de notre co-auteur, le Prof. I. I. Baskin de l'Université d'État de Moscou, consistant à combiner les responsabilités de deux espaces différents durant l'entraînement de la carte, nous l'avons développée et implémentée dans notre logiciel, afin de prédire le profil d'activité d'une molécule sur la base de ses descripteurs ou bien, inversement, prédire des valeurs possibles de descripteurs correspondant à un profil d'activité. Ce nouveau modèle peut être vu comme une porte permettant de passer d'un espace à un autre. Il devient possible de faire des prédictions grâce au passage de l'espace des descripteurs et celui des activités et *vice versa*. En théorie, cette méthode pourrait même être étendue à plus de deux espaces. L'approche S-GTM a

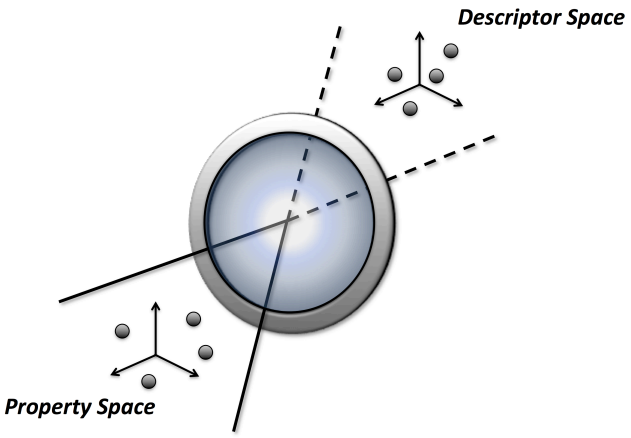


Figure 4: L'approche Stargate GTM permet de voyager de l'espace des descripteurs à celui des propriétés et *vice versa*.

été notamment appliquée à des inhibiteurs de la thrombine et à des complexants de lanthanides, ainsi qu'au profilage de l'affinité de composés ChEMBL pour 8 différentes cibles protéiques (article soumis au *Journal of Chemical Information and Modeling* [5]).

QSAR inverse par GTM et S-GTM

Les relations quantitatives structure à activité (QSAR) inverses permettent de passer, à partir d'activités désirées, à la structure moléculaire ; plusieurs structures peuvent correspondre à un seul profil d'activité. La méthode GTM conventionnelle permet de passer des activités aux descripteurs de diverses manières, la plus simple étant d'utiliser un paysage d'activité ou une superposition de paysages d'activité pour délimiter des zones d'intérêt où de nouveaux composés pourraient être trouvés. Notre nouvelle méthode S-GTM peut également accomplir une telle tâche ; diverses techniques ont été explorées, dont les plus efficaces utilisent de multiples paysages d'activité, ou l'établissement d'une fonction de score basée sur la vraisemblance dans l'espace des descripteurs pondérée par les responsabilités dans l'espace des activités. Durant cette thèse, ces méthodes ont été conçues, implémentées dans un logiciel, et appliquées dans un article en cours de rédaction, dont le sujet est de retrouver des structures moléculaires à partir d'une ou plusieurs affinités pour des cibles protéiques.

Le Logiciel ISIDA/GTM

Un outil en ligne de commande a été programmé pendant la thèse (GTMapTool), associé à une interface graphique (GTMap) et complété par des scripts R et perl pour la visualisation, et dont l'utilisation est décrite dans des manuels et tutoriels pour l'instant

réservés au laboratoire. Les différentes méthodes GTM et S-GTM décrites dans cette thèse ont été implémentées dans ce logiciel, ainsi que les procédures permettant la classification, la régression, l'optimisation de modèles par validation croisée, l'estimation des paramètres statistiques, et les différentes visualisations. L'ensemble de ces outils a été implémenté par l'auteur de la thèse, mis à part le visualiseur de molécules et le logiciel de fragmentation utilisé pour générer les descripteurs.

Conclusion

Les performances de nos modèles prédictifs se rapprochent de celles des méthodes d'apprentissage automatique classiques telles que les machines à vecteurs de support ou les forêts d'arbres aléatoires. Cependant, la plupart de ces méthodes ne fournissent pas de visualisation, et cette dernière peut être essentielle pour tenter de comprendre comment s'organise un univers chimique. Nos modèles ne sont peut-être pas les meilleurs, car l'on perd une certaine quantité d'information en voulant réduire le nombre de dimensions ; cependant, on y gagne en compréhension. Nous avons par exemple utilisé la visualisation GTM pour mieux comprendre l'organisation de l'espace de descripteurs SILIRID (simple ligand-receptor interaction descriptor) et cartographier les interactions protéines-ligands [6]. Pour les grandes librairies chimiques, la méthode iGTM est un outil d'analyse et de visualisation utile, permettant de comprendre pourquoi une librairie est plus concentrée dans telle ou telle région, grâce aux paysages d'activités qui permettent de cartographier n'importe quelle propriété d'un jeu de données. La nouvelle méthode présentée ici, S-GTM, permet non seulement de passer d'un espace à l'autre mais également de voir comment, en essayant de trouver des structures correspondant à une activité ou *vice versa*, on se retrouve parfois devant plusieurs choix possibles. Cette thèse et la boîte à outils qui en a résulté devraient être utiles au chémoinformaticien pour comprendre la structure de son jeu de données.

D'autres pistes en relation avec la cartographie GTM explorées au laboratoire et non étudiées dans cette thèse incluent l'optimisation par algorithme génétique, et la mise en place d'une carte universelle [7] ; l'interaction de l'utilisateur avec différentes visualisations de l'espace chimique serait également une piste de recherche intéressante, et les méthodes de QSAR inverses basées sur une méthode comme la GTM pourraient être améliorées pour obtenir non seulement une estimation des descripteurs, mais également une génération de structures chimiques nouvelles.

Posters

13-26 juin 2014

STRASBOURG SUMMER SCHOOL IN CHEMOINFORMATICS 2014, *GTM Applicability Domain for Regression and Classification Models*, H. A. Gaspar, T. Gimadiev, G. Marcou, A. Varnek.

10-11 octobre 2013

SFCi DAYS, Nancy, *New GTM-Based Applicability Domain for Classification models: Application to the Biopharmaceutics Drug Disposition Classification System (BDDCS)*, H. A. Gaspar, G. Marcou, P. Vayer, A. Varnek.

22-24 juillet 2013

6th JOINT SHEFFIELD CONFERENCE ON CHEMOINFORMATICS, *Generative Topographic Mapping-Based Classification Models and Their Applicability Domain: Application to the BDDCS*, H. A. Gaspar, G. Marcou, P. Vayer, A. Varnek.

25-29 juin 2012

STRASBOURG SUMMER SCHOOL IN CHEMOINFORMATICS 2012, *Generative Topographic Mapping: Universal Tool for Data Visualization, Structure-Activity Modeling and Dataset Comparison* (Best Poster Award), H. A. Gaspar, N. Kireeva, G. Marcou, I. I. Baskin, A. Varnek.

Communications orales

7 novembre 2014

CHEMISTRY PHD STUDENTS DAY, Strasbourg (English), *Visualization of Chemical Space using the GTM*, H. A. Gaspar, D. Horvath, G. Marcou, A. Varnek

12 septembre 2014 10th WORKSHOP IN CHEMICAL INFORMATION, Lausanne (English), *Chemical Data Visualization and Analyis with Incremental GTM: Big Data Challenge*, H. A. Gaspar, I. I. Baskin, G. Marcou, D. Horvath, A. Varnek.

14 avril 2014

SCIENTIFIC DAY OF UMR 7140, Strasbourg (French), *Cartographie de l'espace chimique à l'aide de la méthode GTM*, H. A. Gaspar, D. Horvath, G. Marcou, A. Varnek.

Publications

[1] N. Kireeva, I. I. Baskin, **H. A. Gaspar**, D. Horvath, G. Marcou, A. Varnek. Generative Topographic Mapping (GTM): Universal Tool for Data Visualization, Structure-Activity Modeling and Dataset Comparison, *Molecular Informatics*, 31(3-4):301-312, 2012.

[2] **H. A. Gaspar**, G. Marcou, D. Horvath, A. Arault, S. Lozano, P. Vayer, A. Varnek. Generative Topographic Mapping-Based Classification Models and Their Applicability Domain: Application to the Biopharmaceutics Drug Disposition Classification System (BDDCS), *Journal of Chemical Information and Modeling*, 53(12):3318-3325, 2013.

[3] **H. A. Gaspar**, I. I. Baskin, G. Marcou, D. Horvath, A. Varnek. GTM-Based QSAR Models and Their Applicability Domains, *Molecular Informatics*, 34(6-7):348–356, 2015.

[4] **H. A. Gaspar**, I. I. Baskin, G. Marcou, D. Horvath, A. Varnek. Chemical Data Visualization and Analysis with Incremental GTM: Big Data Challenge, *Journal of Chemical Information and Modeling*, 55(1):84-94, 2015.

[5] **H. A. Gaspar**, I. I. Baskin, G. Marcou, D. Horvath, A. Varnek. Stargate GTM: Bridging Descriptor and Activity Spaces, *Submitted to the Journal of Chemical Information and Modeling*, 2015.

[6] V. Chupakhin, G. Marcou, **H. A. Gaspar**, A. Varnek. Simple Ligand–Receptor Interaction Descriptor (SILIRID) for Alignment-Free Binding Site Comparison, *Computational and Structural Biotechnology Journal*, 10(16):33-37, 2014.

[7] Sidorov, **H. A. Gaspar**, G. Marcou, A. Varnek, D. Horvath, Mappability of Drug-Like Space: Towards a Polypharmacologically Competent Map of Drug-Relevant Compounds, *Submitted to the Journal of Chemical Information and Modeling*, 2015.

Chapter 1

Introduction

The cartography of chemical space aims at visualizing and analyzing the chemical space using maps. Cartography comes from the Ancient Greek χάρτης ("papyrus", "roll", "leaf of paper") and γράφω ("write", "describe", "generate") [8]. The word does not mean the art of making maps, but rather the art of *generating* and *describing* maps, which are representations on a leaf of paper. The *describing* aspect is of paramount importance; the map is not only a visualization tool, but also a way to convey useful and interpretable information for a specific purpose. In the modern world, the sheets of paper have given way to visualizations on computer screens. The cartographers of yesteryear generated maps describing spatial information, with representations of plains, mountains, forests or sees; on the other hand, chemical space cartographers have to map multidimensional descriptor spaces and find a way to highlight their regions of interest. In this thesis, the chemical space is described as a D-dimensional descriptor space, where molecules are characterized by D chemical descriptors. If D is large and the descriptor space impossible to visualize, dimensionality reduction algorithms are used to map molecules into a two or three-dimensional space.

Which existing methods can be used for chemical space cartography?
In Chapter 3, we provide a review of existing dimensionality reduction and visualization techniques for multidimensional spaces, with examples generated by the author of this thesis. We describe *inter alia* the generative topographic mapping approach and its incremental variant designed by C. Bishop and al. [9, 10, 11], as well as the kernel version introduced by Olier et al. [12]; GTM is a non-linear dimensionality reduction algorithm, providing a wide range of visualization possibilities and a probabilistic framework. We used it as a basis for most of our new developments, and to visualize not only individual molecules but also ligand-target interactions (Appendix A) and entire chemical libraries.

What is new in this thesis?

This thesis provides several contributions to chemoinformatics and chemical space visualization:

1. **GTM-based QSAR methods:** we present new GTM-based classification methods, perform GTM-based regression tasks for the first time, and show how to visualize the resulting classification and regression models. We also introduce several new applicability domain definitions.

2. **Methodology for visualizing and comparing chemical libraries:** using probability density functions generated by GTM, an entire chemical library can be represented with a single vector of probabilities. These vectors can be used to evaluate distances between libraries and their dispersion in chemical space. We also introduce the concept of μGTM maps, where libraries are represented by single points.

3. **Methodology for analyzing maps:** GTM-based descriptor or activity landscapes (designed in collaboration with Prof. I. I. Baskin and Prof. A. Varnek) can be used to find regions of interest in chemical spaces; several descriptor scores for GTM maps are also introduced.

4. **A new multispace dimensionality reduction algorithm:** the Stargate generative topographic mapping (S-GTM) is designed for objects "living" in several spaces, such as molecules described in both property and descriptor spaces. Objects can travel from one space to another through the S-GTM "Stargate". S-GTM was developed with the help of Prof. I. I. Baskin and Prof. A. Varnek.

5. **GTM-based inverse QSAR methods**: we present new methods for performing inverse QSAR tasks with both S-GTM and conventional GTM.

6. **Our chemical space visualization software**: we implemented the GTM, kernel GTM and incremental GTM algorithms as well as our own developments into a new software, ISIDA/GTM, comprising a command-line tool (GTMapTool) and a graphical user interface (GTMap). We thank Dr. G. Marcou for his valuable advices on software design.

These developments and their applications are explored in greater depth in the following chapters. Before delving into more practical approaches, let us begin by defining the "chemical space" concept, which is, after all, the primary focus of our research.

Chapter 2

Chemical Space and Chemical Universe

Space comes from the Proto-Indo-European root *spē- "to succeed, prosper, fatten" [13]. There is therefore in the word itself the idea that space "grows fat", stretches beyond its initial limits.

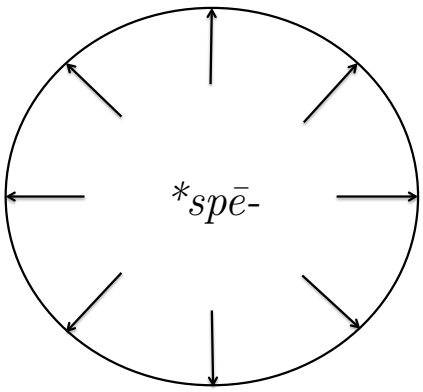


Figure 2.1: Space comes from the Proto-Indo-European root *spē- meaning "to prosper, fatten".

In mathematics, a space is a set with a specific structure. The Oxford Concise Dictionary of Mathematics [14] describes the space as "a set of points with a structure which defines the behaviour of the space and the relationship between the points".

A *chemical space* [15], according to this definition, would be a set of chemical compounds, with a specific structure. Chemical compounds are substances composed of two or more atoms bound by chemical bonds that are characterized by specific properties. An example of chemical space could then be a set of compounds organized in a tree.

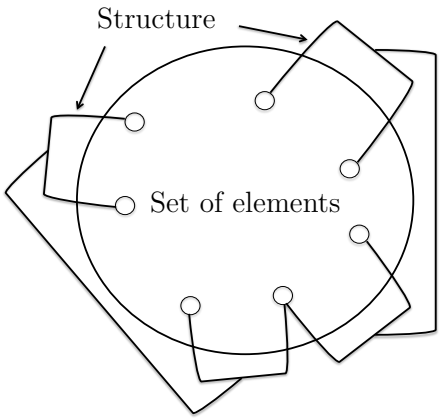


Figure 2.2: In mathematics, a space is a set of some elements with an associated structure.

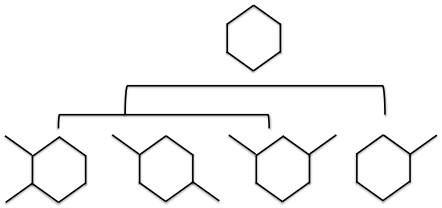


Figure 2.3: A set of compounds (cyclohexane derivates) structured in a tree.

The theoretical chemical space, in its broadest definition, should encompass all possible compounds. However, in practice, we only have access to a more limited set. This set is our *chemical universe*, which changes depending on the task at hand. Some examples of chemical universes include the set of DNA molecules, kinases, small drug-like molecules, etc.

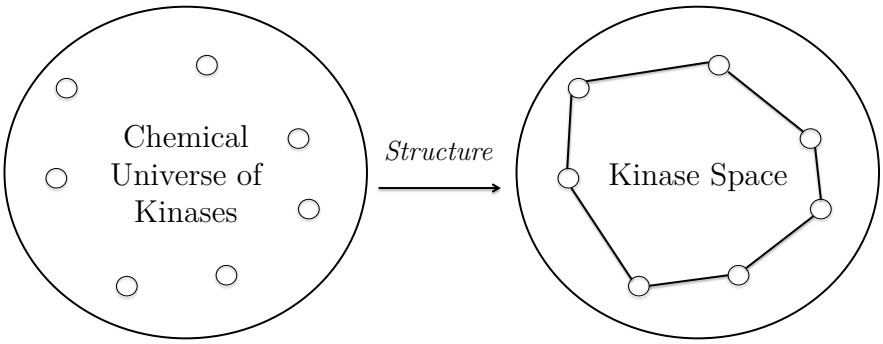


Figure 2.4: The chemical universe of kinases, and an example of kinase space with a structure.

But how can we represent the elements of our chemical universe? Two well-known approaches include representations of molecules as graphs and as sets of descriptors. Chemical descriptors are structural or physico-chemical properties, which can be exper-

imental, or deduced from the 2D or 3D structures. A molecule can therefore be a simple point (vector) in a D-dimensional vector space where each dimension is characterized by a specific descriptor. This type of chemical space then falls into the category of vector spaces, *i.e.*, a closed set of vectors under scalar multiplication and finite vector addition.

If our chemical space is represented as a vector space, we assume that by performing these two basic operations (multiplication by a scalar, and vector addition) on elements (vectors) of our chemical universe we can obtain new points in the chemical space, which might not correspond to any existing chemical compound at all.

Such a space could be based on D chemical descriptors and would therefore be D-dimensional; if it were based on quantum mechanical functions, it could even be infinite-dimensional. In this work, we will use the D-dimensional chemical space based on D structural and/or physico-chemical descriptors as a basis for other representations, and we will name it, as cheminformaticians usually do, the *descriptor space*.

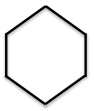
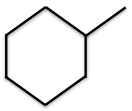
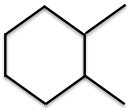
	Mol. Weight	**LogP**	**Polarizability**	**Refractivity**	**...**
	84.16	2.67	11.08	27.61	...
	98.19	2.95	12.93	32.15	...
	112.21	3.24	14.78	36.70	...

Figure 2.5: The descriptor space is a D-dimensional vector space where each molecule is represented by a vector based on properties such as the molecular weight, the octanol/water partition coefficient (logP), polarizability, refractivity, the number of double bonds, the number of oxygens, etc.

For example, if the descriptors of our finite chemical universe are real values, the descriptor space over the field $\mathbb{R}$ will encompass the infinite set of real vectors of length D. If the descriptors are binary, the chemical space over the binary field F2 (the set $\{0,1\}$, together with modulo-2 addition and multiplication [16]) will encompass the finite set of binary vectors of length D.

Instead of representing objects of our chemical universe by an explicit set of D descriptors, it is possible to use the dissimilarities (or similarities) between objects. Sometimes, dissimilarity is the only available data. Kernels, which are inner products between features, therefore similarities, are an analogous representation, but they must

fill Mercer's conditions, *i.e.*, they must be positive and semi-definite. The resulting *dissimilarity space* is also a vector space [17], but with N dimensions, where N is the number of molecules in the chemical universe of interest, and also the number of dissimilarities between one molecule and all others (including itself).

Many other representations of the chemical space are possible, such as the tree represented in Figure 2.3. However, such representations might be hard to visualize when dealing with a large number of compounds. In this thesis, we used the descriptor space approach with real, binary or count data.

From this D-dimensional descriptor space, and working with a given chemical universe of interest, our goal was to produce a 2D map, which could not only be used to visualize compounds as points in a 2-dimensional space and to find clusters (this had already been done before), but also to predict the class or activity of a compound, to visualize predictive models and their applicability domain (AD) and compare libraries of molecules. To meet these goals, we applied the GTM approach [10, 9]; we also employed our new method Stargate GTM (S-GTM) to perform inverse QSAR tasks and predict whole activity profiles. There are several other dimensionality reduction methods, as well as different types of visual structures used in information visualization; we will review most of them in the next chapter, dedicated to existing techniques of multidimensional visualization.

Chapter 3

Visualize a Multidimensional Space

In this chapter, we will review some concepts and techniques linked to the visualization of a multidimensional space: descriptors used in chemistry (corresponding to the dimensions of the space), the concept of intrinsic or hidden dimensionality, some "spooky" phenomena arising in high-dimensional spaces with the infamous curse of dimensionality, and how to visualize the multidimensional space using simple visualization techniques or dimensionality reduction, with some illustrative examples.

3.1 Descriptors

Descriptors are an encoding system for molecules; any value providing a piece of information on a molecule can be considered a descriptor. However, there are many ways to represent a molecule, and many possible descriptors for each molecular model:

1. **The 1D chemical formula.** The simplest molecular representation is the chemical formula, such as CH_2O_2. Descriptors associated with this formula are atomic properties, including atom counts or molecular weight. This representation is not very discriminatory, considering that many molecules are made up of exactly the same atoms.

2. **The 2D molecular graph.** A second possible representation is the molecular graph, corresponding to the *structural formula* in chemistry, wherein a molecule is drawn as a set of vertices (atoms) and edges (bonds). Associated descriptors (2D descriptors) include *inter alia* connectivity information or properties predicted from the 2D structures, such as calculated LogS and LogP.

3. **The 3D molecular structure.** A third possible representation is the 3D molecular structure; the associated descriptors derive from the 3D representation of molecules, for instance the van der Waals volume or the density. A possible 3D

model is the lattice representation, were the 3D structure is embedded in a grid and interacts with a probe. Associated descriptors can be based, for example, on molecular interaction fields (GRID [18], CoMFA [19]), or on 3D pharmacophoric fingerprints.

Another somewhat counter-intuitive classification system discriminates "0D" descriptors (atom counts, bond counts, molecular weight), "1D" descriptors (list of fragments), "2D" descriptors (connectivity), "3D" descriptors (geometry) and "4D" descriptors (3D structure with conformation information). Some descriptors may belong to more than one of these different classes.

In this thesis, we used several sets of chemical descriptors: MACCS keys from MDL Information Systems, ISIDA SMFs (substructural molecular fragments) [20], MOE 2D descriptors [21], or VolSurf+ descriptors [22]. MACCS fingerprints are bit strings encoding the absence or presence of specific structural groups; ISIDA SMFs are 2D descriptors which involve counting the number of occurrences of a specific graph pattern in a molecular structure. MOE 2D descriptors include connectivity information, properties calculated from the 2D structures, or occurrences of specific structural groups; VolSurf+ 3D descriptors are derived from molecular interaction fields generated by the GRID software.

There is no best set of descriptors (yet) to define molecules in all circumstances. However, there might be a best set of descriptors for a given problem. For predictive models, descriptors are generally optimized to achieve the best predictive performance, by using some acquired knowledge on the predicted property (*e.g.*, which features are relevant), and/or by statistical validation (see Chapter 6 on optimization procedures).

3.2 Apparent and Intrinsic Dimensionality

In a descriptor space, the dimensionality is simply defined as the number D of descriptors. The choice of descriptors is always partly arbitrary; we may choose to represent molecules by molecular fragments, experimental properties, and so on. But once we have chosen all our descriptors, we obtain a specific descriptor space with a specific dimensionality.

But is this dimensionality the "real one" for this descriptor space? In other words, do some descriptors convey superfluous information? They often do; in fact, this dimensionality is only apparent.

The intrinsic dimensionality of a dataset, or the minimum number of features needed to describe it given an initial set of descriptors, can be found using diverse methods [23], divided into local and global approaches. The local and global PCA approaches are certainly the easiest to explain and apply. The global PCA approach consists in counting

the number of non-null eigenvalues of the entire data covariance matrix; the local PCA approach or Fukunaga-Olsen's algorithm [24] involves performing a Voronoi tesselation or clustering of the data, and counting the number of normalized eigenvalues of the covariance matrix in each Voronoi region.

Knowing the intrinsic dimensionality might be useful when, for example, performing dimensionality reduction using a method like PCA, which actually discards some descriptor information for visualization; the smaller the intrinsic dimensionality, the lower the risk of losing important information when visualizing a PCA plot. Many dimensionality reduction methods are based on the assumption that the data lies in the proximity of a linear or non-linear manifold of low dimensionality; if the intrinsic dimensionality is small enough, the manifold assumption might not be completely incorrect. However, the instrinsic dimensionality is not a magical number that can be found easily; it is a concept which exists within the framework of the method that defines it. Many conceptually different methods exist and give different estimations; the best policy could be to try some of them and find a consensus value.

Although the intrinsic dimensionality is an important concept that should be kept in mind, when speaking of "dimensionality" we will mainly refer to the *apparent* dimensionality.

3.3 The Curse of Dimensionality

One of the biggest threats a data scientist has to face is the curse of dimensionality. This concept was introduced by Bellman [25] in 1957 while studying dynamic optimization, but is widely used to encompass all the phenomena arising from working with data in a high-dimensional space.

The data scientist's simple desire is to find relationships between data points; however, data points in a high-dimensional space, if not in a sufficient number, may be completely isolated in the considerable volume of the space, which increases exponentially with the number of dimensions. If n points at least are necessary to sample a one-dimensional space, then n^D points will be necessary to sample a D-dimensional space. In practice, we never have that many points to train a model, and this may limit its predictive ability; according to the Hughes Effect, for small datasets, the predictive accuracy of a model decreases as the dimensionality increases beyond the optimal dimensionality; on the other hand, the accuracy increases as the number of points increases. This phenomenon was named after G. Hughes who highlighted it for pattern classifiers in 1968 [26].

A possible representation of a high-dimensional space could be the hypercube. In 3D, most of a uniformly distributed 3D dataset will be near the center of the cube [27];

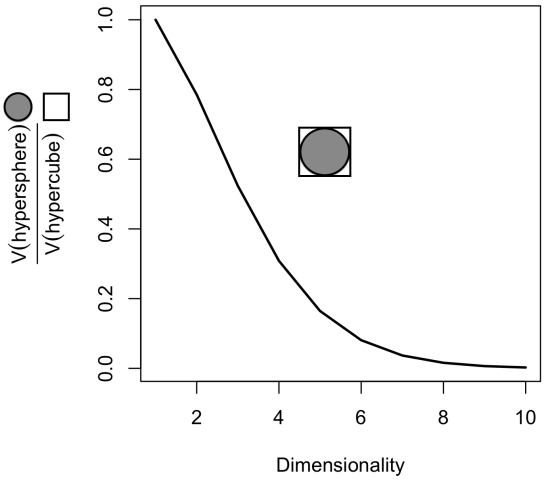


Figure 3.1: Ratio of the volume of the unit hypersphere to the unit hypercube as a function of dimensionality.

however, seemingly opposed to common sense, the data in a hypercube will mainly lie in its corners. Why? Because the volume of a hypercube is concentrated near its corners. This can be observed by plotting the ratio of the volume of the unit hypersphere with radius $r = 1$ to the volume of a unit hypercube with side length $2r$, as shown in Figure 3.1. In 2D, 78% of the unit "hypercube" volume is within the hypersphere; 52% in 3D, and only 4% in 7D, which means than in 7D 96% of the volume is in the corners of the hypercube. With no more than 10 dimensions, there is almost no volume at the center of the hypercube.

Another example of the curse is the nearest neighbors problem in high-dimensional spaces, which has an impact on the performance of k-NN methods. The number of nearest neighbors of a point on a hypergrid grows with the number of dimensions: there are two equivalent nearest neighbors (at the same distance from the point) on a one-dimensional grid, four on a two-dimensional grid, and $2 \times D$ on a D-dimensional grid. Therefore, the concept of neighborhood and similarity may become irrelevant in spaces with a sufficiently high dimensionality.

To avoid the curse or rather diminish its effects, the two most common solutions are feature selection and dimensionality reduction. With feature selection [28, 29], descriptors are selected to decrease the noise due to irrelevant and redundant descriptors, yield a better model generalization and reduce overfitting; for example, the feature selection step can be essential for GWASs (genome-wide association studies), to select the most important single nucleotide polymorphisms associated with specific traits [30, 31, 32]. Feature selection can also be used as a preprocessing step for dimensionality reduction methods. The selection may be supervised (*e.g.*, select descriptors which

correlate the most to a class or activity to be predicted), or unsupervised (*e.g.*, remove descriptors which have a high correlation to each other).

Dimensionality reduction methods, on the other hand, do not merely remove irrelevant descriptors; they use the information contained in all descriptors and find a smaller number of *new*, transformed features; they also provide a way to visualize the data in a reduced 2D or 3D space, if only 2 or 3 of these new features are generated.

3.4 Dimensionality Reduction

When the number of dimensions is too large, it may be useful to use dimensionality reduction methods that will generate a smaller number of new features. These methods provide a way to avoid the dimensionality curse as well as computational costs that may occur when working with high-dimensional data. But to us, their most appreciated quality is the 2D (or 3D) visualization, which provides an overview of the data and a way to find (and visualize) outliers. There are two main types of dimensionality reduction techniques: linear and non-linear. Linear methods usually assume the dataset lives near a low-dimensional *linear* manifold; non-linear methods are used to investigate more complex data structures by assuming that data points lie near a low-dimensional *non-linear* manifold. We will give illustrations of each of the exposed methods using some R and python packages (except for GTM and GTM variants, for which we used our own implementations), always using the same dataset for comparison purposes: 100 inhibitors (red class) and 100 decoys (blue class) of the enzyme acetylcholinesterase, described by ISIDA fragment descriptors (atoms and bonds, length 2-8) [20]. The dataset was randomly selected from the *DUD* database (directory of useful decoys) [33].

3.4.1 Linear Dimensionality Reduction

The goal of dimensionality reduction methods is to project N data points of dimensionality D (data matrix $\mathbf{T}$) into a space with a lower dimensionality L (data matrix $\mathbf{X}$):

$$\mathbf{T} \in \mathbb{R}^{N \times D} \rightarrow \mathbf{X} \in \mathbb{R}^{N \times L} \tag{3.1}$$

Linear dimensionality reduction achieves this goal by applying a linear transformation $\mathbf{U}$ to the N data vectors in D-dimensional space:

$$\mathbf{X} = \mathbf{TU} \tag{3.2}$$

where $\mathbf{U} \in \mathbb{R}^{D \times L}$ is optimized by an objective function. In this section, we will succinctly explore some linear dimensionality reduction methods: principal component analysis (PCA), canonical correlation analysis (CCA), independent component analysis

(ICA), linear discriminant analysis (LDA), linear multidimensional scaling (MDS) and exploratory factor analysis (EFA).

Table 3.1: Some linear dimensionality reduction techniques.

Name	Goal	When to apply
PCA	Minimizes reconstruction error, max. variance	General
CCA	Maximizes correlation between projections	Several spaces
LDA	Maximizes *interclass*/*intraclass* variance	Labeled data
ICA	Maximizes independence of components	Signal separation
MDS	Preserves initial distances	General
EFA	Optimizes factors influencing variables	Identify factors

3.4.1.1 PCA

The most famous linear dimensionality reduction method is principal component analysis or PCA [34]. For PCA, the linear transformation is performed by the matrix $\mathbf{U} \in \mathbb{R}^{D \times L}$ of the eigenvectors of the data covariance matrix with the L highest gl-seigenvalues. The eigenvectors give the directions and eigenvalues the amount of data variance. The objective of the method is to maximize the variance of the projected data, or equivalently minimize the reconstruction error:

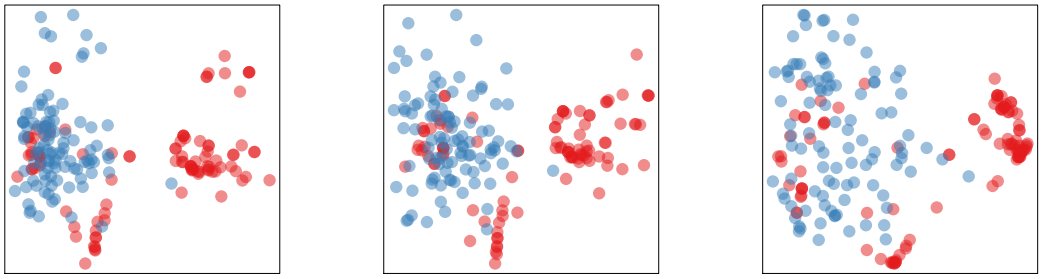
$$E = \left\| \mathbf{T} - \mathbf{X}\mathbf{U}^T \right\|^2 \tag{3.3}$$

where $\mathbf{T}$ is the original data and $\mathbf{X}\mathbf{U}^T$ the reconstructed data with L eigenvectors. A PCA can be performed by first centering the data and then applying a singular value decomposition to find the eigenvectors and eigenvalues of the data covariance matrix. There is no supplementary parameter to tune for PCA.

3.4.1.2 Linear MDS

The goal of metric MDS or metric multidimensional scaling [35, 36] in dimensionality reduction is to obtain a projection which preserves as much as possible the distances between data points in initial space; to achieve this goal, the algorithm minimizes the squared error between distances in initial space $\{d(i,j)\}$ and in the new lower-dimensional space $\{d'(i,j)\}$ measured between the ith and jth objects, or "stress" function. An example of stress function would be:

$$E = \sqrt{\sum_{i<j} (d(i,j) - d'(i,j))^2} \tag{3.4}$$



(a) MDS: Euclidean. **(b)** MDS: Manhattan. **(c)** MDS: Jaccard.

Figure 3.2: MDS projections with different dissimilarity measures: (a) Euclidean (gives the same result as PCA or Kernel PCA), (b) Manhattan, (c) Jaccard. The dataset contains inhibitors (red) and decoys (blue) of acetylcholinesterase, described by ISIDA fragment descriptors.

Metric MDS preserves initial space distances, whereas non-metric MDS only preserves the rank order of distances. MDS with Euclidean distances gives the same results as PCA (or KPCA with a linear kernel). However, any distance measure can be used with MDS. Three examples using different dissimilarity measures are shown in Figure 3.2 (generated in R). There is no supplementary parameter to tune for MDS; however, results may vary depending on the dissimilarity type used to construct the distance matrix. The basic principle of MDS is one of the corner stones of dimensionality reduction, and has been used as a foundation for several dimensionality reduction methods, such as Sammon mapping or SPE [37].

3.4.1.3 CCA

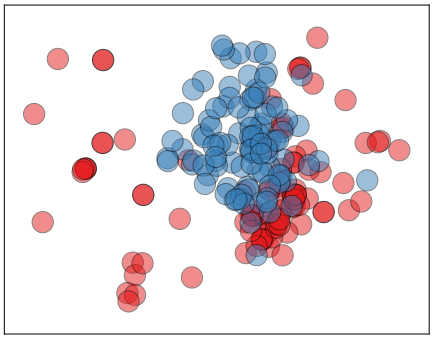


Figure 3.3: CCA (canonical correlation analysis) map example representing inhibitors (red) and decoys (blue) of acetylcholinesterase, optimized using both ISIDA and MOE descriptor spaces.

The goal of canonical correlation analysis or CCA [38] is to maximize the correlation between two linear combinations of variables. Let our data be described in two spaces so that we have the $\mathbf{T_A}$ matrix in Space 1 and the $\mathbf{T_B}$ matrix in Space 2. Then, two transformation will be performed:

$$\mathbf{X}_\mathrm{A} = \mathbf{T}_\mathrm{A}\mathbf{U}_\mathrm{A} \tag{3.5a}$$

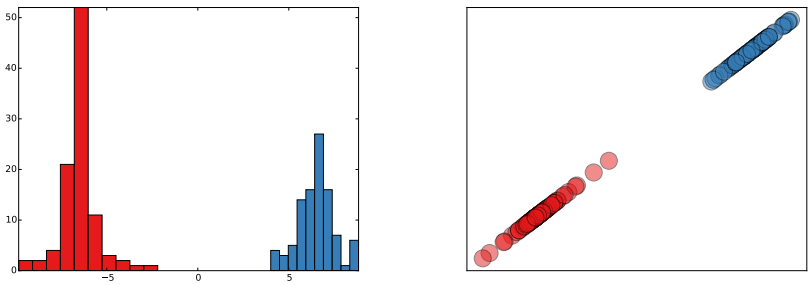
$$\mathbf{X}_\mathrm{B} = \mathbf{T}_\mathrm{B}\mathbf{U}_\mathrm{B} \tag{3.5b}$$

$\mathbf{U}_\mathrm{A}$ and $\mathbf{U}_\mathrm{B}$ are optimized by maximizing the correlation between the two projections:

$$\rho = \max_{\mathbf{U}_\mathrm{A},\mathbf{U}_\mathrm{B}} corr(\mathbf{T}_\mathrm{A}\mathbf{U}_\mathrm{A}, \mathbf{T}_\mathrm{B}\mathbf{U}_\mathrm{B}) \tag{3.6}$$

$\mathbf{X}_\mathrm{A}$ and $\mathbf{X}_\mathrm{B}$ are called the canonical variates. This algorithm is useful when objets are described in two different spaces, *e.g.*, molecules described in two descriptor spaces or molecules described in an experimental activity space and a descriptor space. The visualization of individual objects can be made by plotting the coordinates of the L first canonical variates; for scatterplots, if $L = 2$, axes will be defined by $\mathbf{x}_\mathrm{A}^1$ and $\mathbf{x}_\mathrm{A}^2$ (or $\mathbf{x}_\mathrm{B}^1$ and $\mathbf{x}_\mathrm{B}^2$). This method does not require to tune any other parameter. We provide an example of CCA dimensionality reduction for acetylcholinesterase using two spaces (Figure 3.3, generated with *scikit-learn* [39]): the space of ISIDA descriptors (atoms and bonds, length 2-8) and the space of 186 MOE 2D descriptors.

3.4.1.4 LDA



(a) LDA (barplot). **(b)** LDA (scatter plot).

Figure 3.4: One-dimensional LDA map of inhibitors (red) and decoys (blue) of acetylcholinesterase, described by ISIDA fragment descriptors: (a) bar plot visualization (b) scatterplot visualization.

Linear discriminant analysis or LDA [40, 41] can be used for classification tasks, with

labeled data. LDA maximizes the $\frac{\text{interclass variance}}{\text{intraclass variance}}$ ratio, and projects the data into an L-dimensional space with $L = N_C - 1$, where N_C is the number of classes. And example of LDA for a dataset with two classes (inhibitors and decoys of acetylcholinesterase) is given in Figure 3.4 (generated with *scikit-learn* [39]); the model can be represented as a histogram since for two classes $L = 2 - 1 = 1$. LDA can also be used as a supervised linear classifier, and has no supplementary parameter to tune.

3.4.1.5 ICA

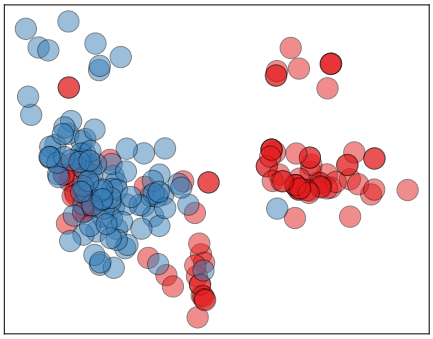


Figure 3.5: ICA (Independent Component Analysis) map example representing inhibitors (red) and decoys (blue) of acetylcholinesterase, described by ISIDA fragment descriptors.

Independent component analysis or ICA [42] is mainly used in blind signal separation but can also be employed for dimensionality reduction. A typical case of blind signal separation is the "cocktail party problem", where individual voices engaged in simultaneous conversations have to be differentiated, *i.e.*, the source signals have to be retrieved. ICA is dependent on a random initialization but otherwise has no parameter to tune. An example of visualization is given in Figure 3.5 (made with *scikit-learn* [39]).

3.4.1.6 EFA

Exploratory factor analysis or EFA is used to build linear generative latent variable models, by means of a linear mapping with Gaussian noise [43], so that, for a data matrix $\mathbf{T}$:

$$\mathbf{T} - \mathbf{M} = \mathbf{X}\mathbf{U}^T + \mathbf{E} \tag{3.7}$$

where $\mathbf{X}$ are the latent variables or factors, $\mathbf{U}$ the factor loading matrix, $\mathbf{E}$ Gaussian noise and $\mathbf{M}$ the bias. Each factor loading's square is the percent of variance of a variable explained by a factor. The factor loadings can be fitted by maximum likelihood. EFA gives results close to PCA, but its philosophy is quite different: in PCA, the initial

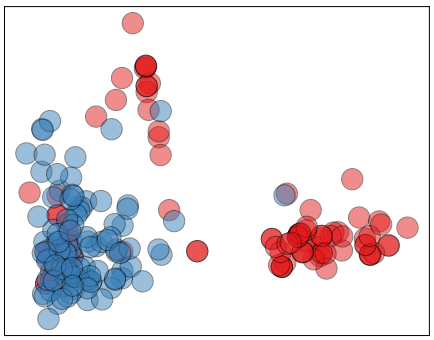


Figure 3.6: EFA (exploratory factor analysis) map example representing inhibitors (red) and decoys (blue) of acetylcholinesterase, described by ISIDA fragment descriptors.

data is represented by a linear combination of factors (descriptive model), whereas in FA, the factors are represented as a linear combination of initial variables (generative model). In PCA, all the data variance is analyzed, whereas in EFA only the *common* variance is accounted for. We show here an EFA example (Figure 3.6) obtained with *scikit-learn* [39]. EFA is to be distinguished from CFA (confirmatory factor analysis), which aims at confirming a hypothesized structure instead of finding the latent structure. Another family of dimensionality reduction methods related to factor analysis is NNMF (non-negative matrix factorization) [44] for non-negative input matrices, which uses non-negativity constraints.

3.4.2 Non-Linear Dimensionality Reduction

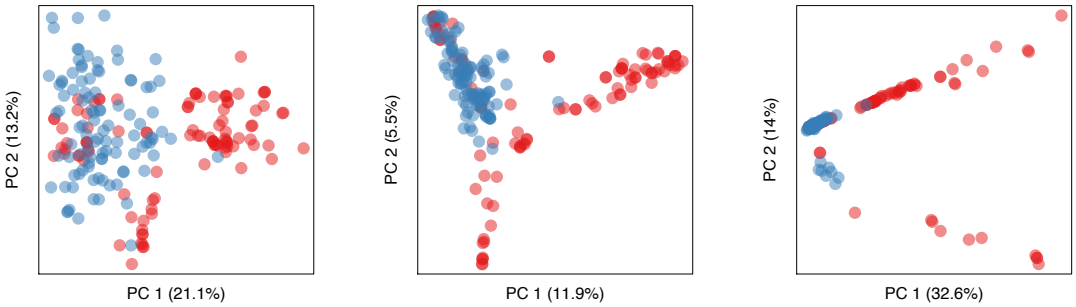
Table 3.2: Some non-linear dimensionality reduction techniques.

Name	**Key concept**
KPCA	Performs an eigendecomposition of a kernel
Sammon	Preserves initial distances, favors small distances
Isomap	Preserves geodesic distances
LLE	Performs linear local approximations
Laplacian Eigenmaps	Performs an eigendecomposition of the graph Laplacian
t-SNE	Minimizes divergence between initial and latent distributions
Autoencoder	Learns to reconstruct the data
SOM	Topology-preserving neural network
GTM	Fits iteratively a manifold to the data, probabilistic SOM

In this section, we will review some non-linear dimensionality reduction techniques shown in Table 3.2, including KPCA, Sammon mapping, Isomap, locally linear embedding (LLE), Laplacian Eigenmaps, autoencoders, self-organizing maps (SOMs) and

generative topographic mapping (GTM). Sammon mapping is an MDS variant; Isomap, LLE and Laplacian Eigenmaps are based on the construction of a k-NN graph and are closely related to KPCA; t-SNE is a distance-preserving, probabilistic method; Autoencoders, SOM and GTM are based on neural networks. For all these dimensionality reduction techniques, the underlying assumption is that the data lies on a low-dimensional non-linear manifold embedded in the D-dimensional descriptor space. Most of them rely on preserving the neighborhood of points, and aim at unrolling the manifold to obtain a lower-dimensional visualization.

3.4.2.1 KPCA



(a) PCA. **(b)** KPCA: RBF kernel. **(c)** KPCA: polynomial kernel.

Figure 3.7: (a) Conventional PCA, compared to kernel PCA maps with (b) an RBF kernel and (c) a polynomial kernel. The dataset contains inhibitors (red) and decoys (blue) of acetylcholinesterase, described by ISIDA fragment descriptors.

Kernel PCA or KPCA [45] is very similar to PCA; however, instead of working with a data matrix $\mathbf{T}$ of N instances and D dimensions, PCA operations are performed on an $N \times N$ kernel $\mathbf{K}$ that has been previously centered. The kernel matrix can be considered as a similarity matrix which is positive semi-definite and symmetric. The kernel approach allows us to work in an implicit feature space via the kernel trick: data points $\mathbf{t}$ are mapped into a feature space of higher dimensionality with a function $\phi(\mathbf{t})$ that is never computed; instead, the inner product between objects in feature space is reduced to a kernel function in input space:

$$K(\mathbf{t}_i, \mathbf{t}_j) = \phi(\mathbf{t}_i)\dot{\phi}(\mathbf{t}_j) \tag{3.8}$$

which results in an $N \times N$ kernel $\mathbf{K}$ characterizing objects in the possibly infinite-dimensional feature space. In other words, the goal is to map the data into a high-dimensional space using a kernel function. The reason for using the kernel trick is that it is easier to separate linearly objects in a space with a higher dimensionality; therefore,

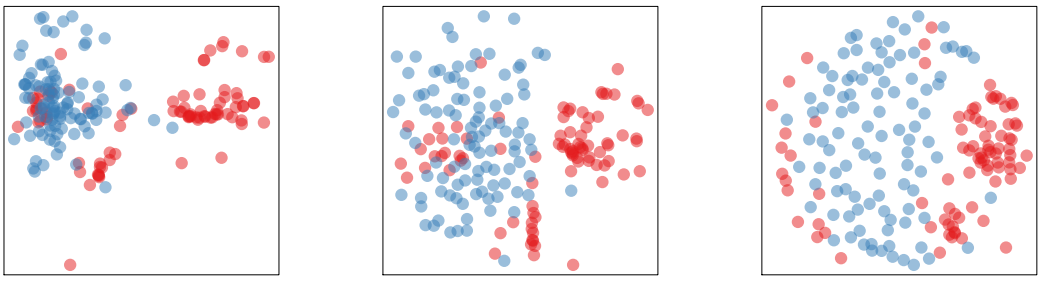
this approach is particularly interesting for non-linear clustering (*e.g.*, with k-means). Examples of KPCA maps (obtained with the R package *kernlab* [46]) for inhibitors and decoys of acetylcholinesterase, described by ISIDA fragment descriptors, are given in Figure 3.7; RBF and polynomial kernels were used; the γ parameter regulating the width of the RBF kernel was set to 0.2. It should be noted that a PCA model can been seen as a KPCA model with a linear kernel [45] computed from centered data. Depending on the kernel, different parameters should be tuned. For the RBF kernel, it is the γ parameter:

$$K(\mathbf{t}_i, \mathbf{t}_j) = \exp\left(-\gamma \left\|\mathbf{t}_i - \mathbf{t}_j\right\|^2\right) \tag{3.9}$$

For the polynomial kernel, the user can tune the degree d, a constant *cste* and eventually the slope a:

$$K(\mathbf{t}_i, \mathbf{t}_j) = \left(a\mathbf{t}_i^T\mathbf{t}_j - cste\right)^d \tag{3.10}$$

3.4.2.2 Sammon Mapping



(a) Sammon: Euclidean. **(b)** Sammon: Manhattan. **(c)** Sammon: Jaccard.

Figure 3.8: Sammon maps with different dissimilarities: (a) Euclidean, (b) Manhattan, (c) Jaccard. The dataset contains inhibitors (red) and decoys (blue) of acetylcholinesterase, described by ISIDA fragment descriptors.

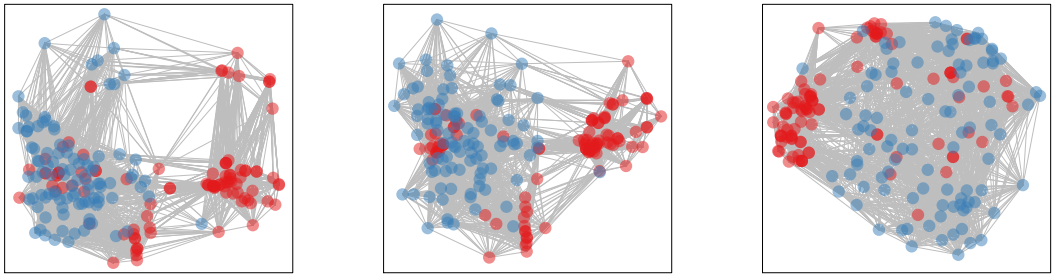
Sammon mapping is a non-linear variant of MDS; it is the same algorithm, except that the stress function to be minimized is normalized by the initial space distances [47]:

$$E = \frac{1}{\sum_{i<j} d(i,j)} \sum_{i<j} \frac{(d(i,j) - d'(i,j))^2}{d(i,j)} \tag{3.11}$$

where $\{d(i,j)\}$ are the distances in initial space and $\{d'(i,j)\}$ the distances in the reduced space measured between the ith and jth objects. By dividing by the initial space distances, the optimization favors small distances (larger stress function) over large distances; as the non-linearity in data is better approximated by smaller distances, the method integrates some non-linear information in this way. Three examples using differ-

ent dissimilarity measures are used in Figure 3.8 (made with the R package *MASS* [48]). As MDS, Sammon mapping does not require a parameter optimization, but results may vary depending on the chosen dissimilarity type.

3.4.2.3 Isomap



(a) Isomap: Euclidean. **(b)** Isomap: Manhattan. **(c)** Isomap: Jaccard.

Figure 3.9: Isomaps with 30 neighbors and different dissimilarity measures: (a) Euclidean, (b) Manhattan, (c) Jaccard. The dataset contains inhibitors (red) and decoys (blue) of acetylcholinesterase, described by ISIDA fragment descriptors.

Isomap [49, 50] is also an MDS-derived method, which uses geodesic distances from neighborhood graphs, and learns the global geometry of a dataset by using local metric information. Isomap can be seen as special case of KPCA with geodesic distances as kernel. The Isomap algorithm consists in three steps:

1. Retain only dissimilarities of k nearest neighbors for each object and draw a graph where all data points are connected to their nearest neighbors (k-NN graph); the weight of edges between neighboring vertices (objects) i and j is equal to the initial space distance $d(i, j)$.

2. Compute shortest path distances (matrix $\mathbf{G}$) between all points or "geodesic" distances minimizing the sum of weights of edges ($d(i, j)$) between the graph's vertices, with, *e.g.*, Dijkstra's algorithm for undirected graphs [51].

3. Perform a classical MDS preserving the geodesic distances $\mathbf{G}$.

The Isomap's output we are interested in are the L eigenvectors with best eigenvalues of the geodesic distance matrix $\mathbf{G}$. The Isomap model depends on the number of neighbors k chosen to build the k-NN graph. Three examples using different dissimilarity measures are used in Figure 3.9, with 30 neighbors (made with the *vegan* R package [52]).

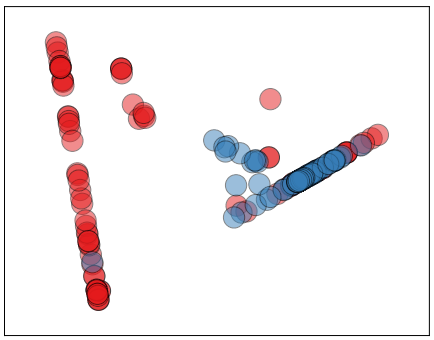


Figure 3.10: LLE (locally linear embedding) map example representing inhibitors (red) and decoys (blue) of acetylcholinesterase, described by ISIDA fragment descriptors.

3.4.2.4 LLE

Locally linear embedding or LLE [53] computes an embedding preserving the neighborhood of data points. LLE assumes that data points in the initial space and their neighbors are close to a locally linear patch of manifold. The dimensionality reduction is performed in 3 steps:

1. The k nearest neighbors $\{\mathbf{t}_k\}$ of each point $\mathbf{t}_i$ are computed (k-NN graph).

2. The weights $\{V_{ik}\}$ that best describe the data points $\{\mathbf{t}_i\}$ as a combination of their neighbors $\{\mathbf{t}_k\}$ are found by minimizing the following cost function:

$$C(\mathbf{V}) = \sum_i \left| \mathbf{t}_i - \sum_k V_{ik}\mathbf{t}_k \right|^2 \tag{3.12}$$

3. Then, the projection to the L-dimensional space is done by finding the low-dimensional coordinates of data points $\{\mathbf{x}_i\}$ (with neighbors $\{\mathbf{x}_k\}$) minimizing the following cost function:

$$C(\mathbf{X}) = \sum_i \left| \mathbf{x}_i - \sum_k V_{ik}\mathbf{x}_k \right|^2 \tag{3.13}$$

Therefore, the data points can be reconstructed from a combination of their neighbors using the same weights in the initial or low-dimensional space. As Isomap, LLE can be seen as a KPCA variant, and has a parameter k (the number of nearest neighbors used to construct the k-NN graph) that must be tuned. An example of the method as implemented in *scikit-learn* [39] is given in Figure 3.10, using 30 neighbors.

3.4.2.5 Laplacian Eigenmaps

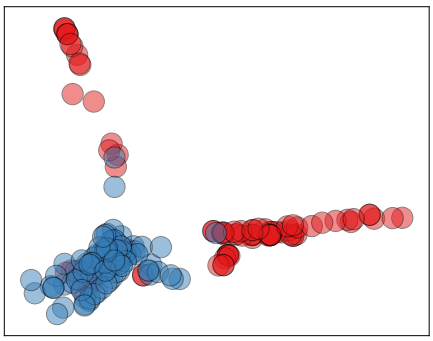


Figure 3.11: Laplacian Eigenmaps example representing inhibitors (red) and decoys (blue) of acetylcholinesterase, described by ISIDA fragment descriptors.

As LLE and Isomap, the Laplacian Eigenmaps technique [54] begins by constructing a neighborhood graph (k-NN graph). It builds a weighted adjacency matrix with a heat kernel or binary weights, and the embedding is computed from the normalized Laplacian. The algorithm takes 3 steps:

1. The k nearest neighbors $\mathbf{t}_k$ of each point $\mathbf{t}_i$ are computed (k-NN graph).

2. The weights $\mathbf{V}$ are computed, using a heat kernel:

$$V_{ij} = \exp\left(-\gamma \left\|\mathbf{t}_i - \mathbf{t}_j\right\|^2\right) \tag{3.14}$$

   where γ is a tunable parameter. Weights V_{ij} can also be set as equal to 1 if $\mathbf{t}_i$ and $\mathbf{t}_j$ are connected in the k-NN graph and equal to 0 otherwise.

3. The projection of points into the lower-dimensional space is performed by the eigen-decomposition of the graph Laplacian:

$$\mathbf{LU} = \mathbf{lDU} \tag{3.15}$$

   where $D_{ii} = \sum_j V_{ij}$ is the degree of vertex i, $\mathbf{L}$ is the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{V}$, $\mathbf{l}$ the eigenvalues and $\mathbf{U}$ the eigenvectors used for the embedding in the space of lower dimensionality.

Laplacian Eigenmaps, as LLE and Isomap, could therefore be interpreted as a variant of KPCA, with a tunable parameter k, the number of neighbors to build the k-NN graph; another parameter has to be taken into account (γ) if the heat kernel is chosen.

An example of dimensionality reduction with Laplacian Eigenmaps as implemented in *scikit-learn* [39] is given in Figure 3.11.

3.4.2.6 t-SNE

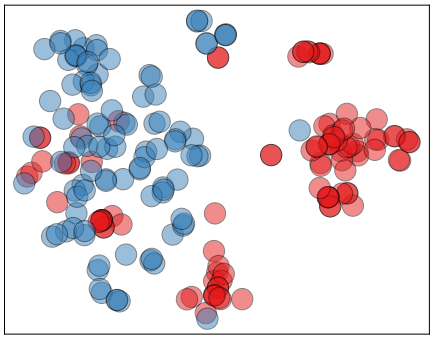


Figure 3.12: t-SNE (t-distributed stochastic neighbor embedding) example representing inhibitors (red) and decoys (blue) of acetylcholinesterase, described by ISIDA fragment descriptors.

t-SNE or t-distributed stochastic neighbor embedding [55], a variant of SNE [56] (stochastic neighbor embedding), is an efficient method for datasets which have more than one embedded manifold; it focuses on local data structures, and is a good choice for separating clusters. As many MDS-related methods, the goal of SNE and t-SNE is to preserve dissimilarities in the original data.

Conventional SNE measures Gaussian joint probabilities in the original space so that the probability p_{ij} is proportional to the similarity between points $\mathbf{t}_i$ and $\mathbf{t}_j$; Gaussian "induced" probabilities q_{ij} are also measured for each pair of points in low-dimensional space $\mathbf{x}_i$ and $\mathbf{x}_j$. In other words, the similarities between points in the initial space and in the latent space are translated into probabilities. The position of points in latent space is updated by minimizing the Kullback-Leibler divergence [57] between joint probabilities in the input and latent space using gradient descent; the KLB divergence can be related in this case to the MDS error function.

In t-SNE, the probabilities q_{ij} between each pair of points in low-dimensional space are computed using a Student t-distribution instead of a Gaussian, so that points would not be gathered at the same place (crowding effect): the Student t-distribution has heavy tails that allow to represent moderate distances in the initial space by larger distances on the 2-dimensional map. t-SNE also uses a symmetrized version of the cost function. SNE and t-SNE have several parameters: the perplexity of Gaussian distributions in the initial high-dimensional space, the learning rate, and eventually the

early exaggeration, a multiplication factor for probabilities in the initial space at the early stages of the optimization process. The perplexity can be related to a measure of the number of neighbors (cf. Isomap); the learning rate used for gradient descent has a great impact on the shape of the map, and the early exaggeration improves the separation of clusters. We show a t-SNE example in Figure 3.12, computed with the *scikit-learn* [39] implementation, with learning rate = 100, perplexity = 30, and early exaggeration = 6.

3.4.2.7 Autoencoders

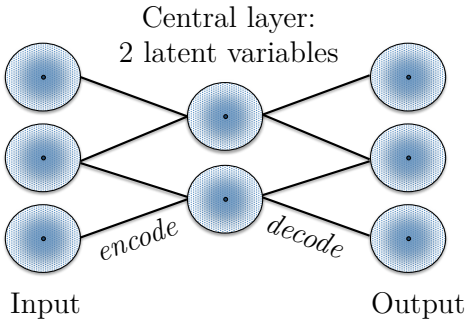


Figure 3.13: Autoencoders are symmetric multilayer neural networks that learn to reconstruct the data; in this diagram, 2 central hidden units are used to reconstruct the 3 initial variables.

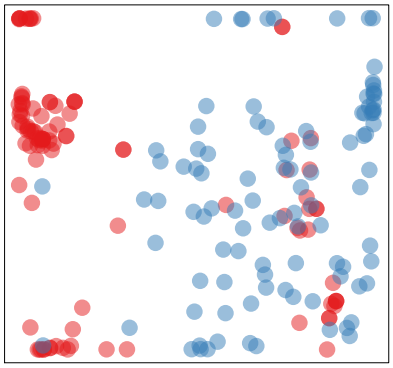


Figure 3.14: Autoencoder dimensionality reduction example representing inhibitors (red) and decoys (blue) of acetylcholinesterase, described by ISIDA fragment descriptors.

Autoencoders [58] are symmetric multilayer neural networks with a small central layer trained to encode and reconstruct the initial dataset (Figure 3.13). In other words, the autoencoder learns the identity matrix. With the small central layer, the net automatically finds a small number of hidden features in the dataset. The autoencoder

uses backpropagation to adjust its weights and to optimize its cost function so that its input (initial data) becomes as similar as possible to its output. The autoencoder is an attractive method, very simple to understand, but with several parameters to tune. In Figure 3.14 we show an example of dimensionality reduction performed by the simplest type of network (implemented in the *autoencoder* R package [59]) with only 3 layers (input, hidden, output), and 2 hidden units corresponding to the number of "latent" dimensions. This network, with 3 layers and 2 hidden units with logistic activation functions depends on four other parameters: the random weight initialization (here random values drawn from $\mathcal{N}(\mu = 0, \sigma^2 = 0.001)$), a weight decay parameter or regularization coefficient λ (here set to 2.10^{-4}), and, since we used a sparse autoencoder, two other parameters for the sparsity: the weight of the sparsity penalty term in the autoencoder objective (set to 6), and the sparsity parameter (set to 0.5).

3.4.2.8 SOM

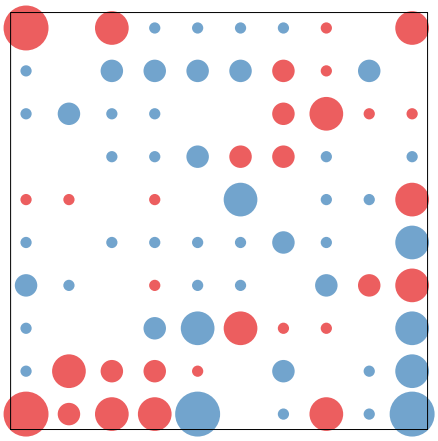


Figure 3.15: Self-organizing map (SOM) example representing the density of inhibitors (red) and decoys (blue) of acetylcholinesterase at each node of the SOM grid by the node size.

Like Autoencoders, Kohonen maps or self-organizing maps (SOMs) [60] are neural networks, inspired by the structure of biological neurons. The particularity of SOM comparing to other neural networks is the conservation of the initial space topology; instances closer in the initial space will be closer on the 2D map. The algorithm begins by arranging artificial "nodes" on the 2D map, usually on a regular grid; to each node is associated a weight vector of dimension D, the dimension of the initial space. Distances can then be computed between the nodes' weight vectors and the instances in initial space; the node closest to each data instance is retrieved, and its weights as well as those of its neighbors are updated to move closer to the data instance. The final position of a data point on the 2D map will be at the same position as its best matching node.

As represented on Figure 3.15 showing a SOM map computed with the *kohonen* [61] package, the representation is usually that of a regular grid; different ways were designed to better visualize the proportion of data points mapped at each node; here, we only represent higher populations by a higher color intensity and color each node by the major class. SOM has 4 main parameters: the learning rate, the neighborhood size, the strategy used to modify the learning rate and neighborhood size as the training progresses, and the resolution (number of nodes) (in Figure 3.15, set to 10×10). The SOM can be randomly initialized (this is the case in Figure 3.15), or the weights can be chosen from the principal components (PCA initialization).

3.4.2.9 GTM

GTM [10, 9] is the probabilisitc counterpart of SOM; it is a generative latent variable model (it assumes that data is generated from a small number of latent variables), and is also non-linear, with a strong probabilistic basis, and has a log likelihood objective function which is optimized during training. It depends on 4 parameters: the map resolution or number of nodes K, the number of RBF functions M, a factor influencing the width of RBF functions w, and a regularization coefficient λ. This method provided us with a probabilistic framework, making it possible to build classification and regression models and to perform various other tasks such as chemical database analysis or inverse QSAR-like approaches. Its main drawback is a large number of parameters to tune. The method is explained in further detail in the next section.

3.4.3 Visual Separation of Classes

For information purposes, we provide the balanced accuracy of all the plots we have shown previously (Table 3.3), using the labeled dataset of 100 inhibitors and 100 decoys of acetylcholinesterase described by ISIDA descriptors (atoms and bonds, length 2 to 8). To see how the balanced accuracy is computed, see Chapter 6; the BAC value can only take values between 0 and 1; the model is inefficient for $BAC < 0.5$ and classifies instances perfectly when $BAC = 1$. Here, we don't use the balanced accuracy as an actual performance measure but as an indicator of the visual separation of classes on our fitted maps. Therefore, this is not in any way a comparative study of classification performances of all the previously shown techniques; no optimization of parameters was performed. We attributed a "predicted" class to each 2D point by finding the most representative class among its k nearest neighbors (k-NN classification), and compared this fitted prediction to the experimental classes of these compounds. Most of these maps show a good separation, especially SOM, GTM, kernel PCA with an RBF kernel and t-SNE.

Table 3.3: Visual separation of classes on the maps shown in this chapter, measured by balanced accuracy BAC_{FIT}.

Method	**Figure**	BAC_{FIT}
MDS (Euclidean), PCA	Figure 3.2a	0.79
MDS (Jaccard)	Figure 3.2c	0.77
MDS (Manhattan)	Figure 3.2b	0.83
ICA	Figure 3.5	0.87
CCA	Figure 3.3	0.87
EFA	Figure 3.6	0.79
KPCA (RBF)	Figure 3.7b	0.90
KPCA (Polynomial)	Figure 3.7c	0.84
Sammon (Euclidean)	Figure 3.8a	0.76
Sammon (Jaccard)	Figure 3.8c	0.60
Sammon (Manhattan)	Figure 3.8b	0.66
Isomap (Euclidean)	Figure 3.9a	0.79
Isomap (Jaccard)	Figure 3.9c	0.76
Isomap (Manhattan)	Figure 3.9b	0.78
Locally Linear Embedding	Figure 3.10	0.79
Laplacian Eigenmaps	Figure 3.11	0.83
t-SNE	Figure 3.12	0.91
Autoencoder	Figure 3.14	0.80
SOM	Figure 3.15	0.92
GTM	Figure 3.17a	0.90
KGTM (RBF)	Figure 3.17b	0.89
KGTM (Polynomial)	Figure 3.17c	0.87

3.5 Generative Topographic Mapping

The generative topographic mapping or GTM [10, 9] was introduced by Bishop et al. in the 1990s. It provides a probabilistic framework for Kohonen maps and guarantees convergence, which can be checked with an objective function. GTM is a manifold-based non-linear dimensionality reduction method, for which there is a topological ordering: points on the manifold in the low-dimensional space will also be close to their mappings in high-dimensional space. The method is "generative", for it is assumed that the D dimensions are generated by a smaller number of latent or hidden variables. The probabilistic framework of GTM allowed us to build regression and classification models and create different types of data visualization. There are many variants of GTM; for small datasets, the original algorithm is sufficient; however, for large amounts of data, the incremental version (iGTM), also described by Bishop [11], is a valuable solution. For data in similarity space, the kernel algorithm was introduced by Olier and al. [12]; an LTM algorithm was introduced by Kaban and al. to deal with binary or count data [62], and was applied by Owen and al. to the visualization of molecular fingerprints [63]; finally, we introduce in this thesis a multispace version of GTM or "Stargate GTM" in Chapter 11, which can be used as a multi-space dimensionality reduction method (as CCA), but also as a supervised regression technique and for navigating between different descriptor spaces. In this section, we use most of the time the same symbols as in Bishop's article.

3.5.1 Original GTM Algorithm

GTM performs dimensionality reduction from the initial D-dimensional descriptor space to the viewable 2-dimensional latent space. A regular grid of K nodes covering the 2-dimensional space is generated, and each node $\mathbf{x}_k$ is mapped to a manifold point $\mathbf{y}_k$ embedded in the D-dimensional space: $\mathbf{x}_k \rightarrow \mathbf{y}_k$ (Figure 3.16), using the function $y(\mathbf{x}; \mathbf{W})$ that maps points from the two-dimensional latent space into the D-dimensional data space:

$$y(\mathbf{x}; \mathbf{W}) = \mathbf{W}\phi(\mathbf{x}) \tag{3.16a}$$

$$\mathbf{Y} = \mathbf{W}\mathbf{\Phi}^{\mathrm{T}} \tag{3.16b}$$

where $\mathbf{Y}$ is the $K \times D$ manifold, $\mathbf{W}$ is the $D \times M$ parameter matrix, and $\mathbf{\Phi}$ is the $M \times K$ radial basis function matrix with M RBF centers $\boldsymbol{\mu}_m$:

$$\Phi_{mk} = \exp\left(\frac{\left\|\mathbf{x}_k - \boldsymbol{\mu}_m\right\|^2}{2\sigma^2}\right) \tag{3.17}$$

In our implementation, σ^2 is initialized as the average squared Euclidean distance τ between two RBF centers multiplied by a tunable factor w, and $\mathbf{W}$ is initialized as follows:

$$\mathbf{W} = \mathbf{U}\mathbf{X}^T\mathbf{\Phi}^{-1} \tag{3.18}$$

where $\mathbf{U}$ is a matrix containing the two first eigenvectors and $\mathbf{X}$ the $K \times 2$ matrix comprising K grid nodes; $\mathbf{W}$ is initialized in this way to minimize the sum-of-squares error between the GTM mapping of latent points and the PCA mapping (cf. Bishop's article [9] for further detail); β^{-1} is initialized as the third PCA eigenvalue. Therefore, in order to correctly position the manifold in the initialization step, a PCA step has to be performed. The points $\mathbf{y}_k$ on the manifold $\mathbf{Y}$ are the centers of normal probability distributions (NPD) of $\mathbf{t}$:

$$p(\mathbf{t}|\mathbf{x}_k, \mathbf{W}, \beta) = \frac{\beta}{2\pi}^{D/2} \exp\left(-\frac{\beta}{2}\|\mathbf{y}_k - \mathbf{t}\|^2\right) \tag{3.19}$$

where $\mathbf{t}_n$ is a data instance and β is the common inverse variance of these distributions. An optimal GTM map corresponds to the highest log likelihood $\mathcal{L}(\mathbf{W}, \beta)$, optimized by EM (expectation-maximization) [64]. The formula for the complete log likelihood is:

$$\mathcal{L}(\mathbf{W}, \beta) = \sum_n ln\left\{\frac{1}{K}\sum_k p(\mathbf{t}_n|\mathbf{x}_k, \mathbf{W}, \beta)\right\} \tag{3.20}$$

β and $\mathbf{W}$ are optimized during the maximization step:

$$\frac{1}{\beta} = \frac{1}{ND}\sum_n\sum_k R_{kn}\|\mathbf{y}_k - \mathbf{t}_n\|^2 \tag{3.21}$$

$$\left(\mathbf{\Phi}^T\mathbf{G}\mathbf{\Phi} + \frac{\lambda}{\beta}\right)\mathbf{W}^T = \mathbf{\Phi}^T\mathbf{R}\mathbf{T} \tag{3.22}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{G}$ a $K \times K$ matrix with elements $G_{kk} = \sum_n R_{kn}$. The four GTM parameters that can be tuned by the user are the number M of RBFs, the number of nodes K, a multiplication factor w for the RBF width (Figure 3.16), as well as the regularization parameter λ for the weight matrix $\mathbf{W}$, which is used to avoid overfitting and can be seen as a smoothness parameter for the manifold. In our implementation, we place RBF centers $\{\boldsymbol{\mu}_m\}$ and nodes $\{\mathbf{x}_k\}$ on two square grids in the 2D space; the parameters set by the user are actually $\sqrt{M}$ and $\sqrt{K}$, to obtain $\sqrt{M} \times \sqrt{M}$ and $\sqrt{K} \times \sqrt{K}$ grids. The responsibility or posterior probability that the given point $\mathbf{t}_n$ in the data space is generated from the kth node is computed using

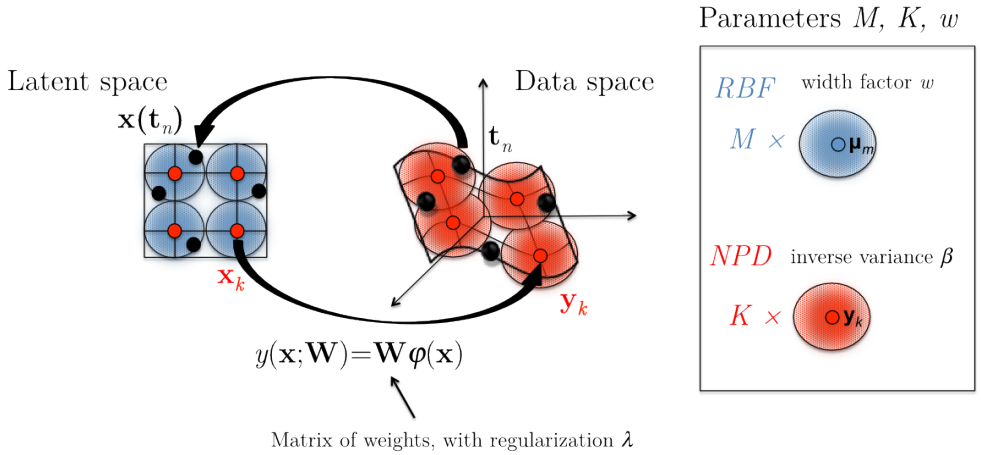


Figure 3.16: GTM concept and its parameters M (number of RBFs or radial basis functions), K (number of NPDs or normal probability distributions), w (RBF width factor) and λ (weight regularization coefficient). The RBFs and NPDs are symbolized by the large blue circles and red spheres, respectively. Each node $\mathbf{x}_k$ in the latent space is mapped to its position $\mathbf{y}_k$ in the descriptor space, using an ensemble of RBFs $\mathbf{\Phi}$ (with centers $\boldsymbol{\mu}_m$) and a weight matrix $\mathbf{W}$. The $\mathbf{y}_k$ points are the centers of the NPDs, from which are computed the responsibilities. These responsibilities are used as weighting factors to map the data points from the initial space $\mathbf{t}_n$ to their position $\mathbf{x}(\mathbf{t}_n)$ in 2D.

Bayes' theorem:

$$R_{kn} = p(\mathbf{x}_k|\mathbf{t}_n, \mathbf{W}, \beta) = \frac{p(\mathbf{t}_n|\mathbf{x}_k, \mathbf{W}, \beta)p(\mathbf{x}_k)}{\sum_{k'=1}^{K} p(\mathbf{t}_n|\mathbf{x}_{k'}, \mathbf{W}, \beta)p(\mathbf{x}_{k'})} \tag{3.23}$$

These responsibilities $\mathbf{R}$ are used to compute the mean position of a molecule on the map $\mathbf{x}(\mathbf{t}_n)$, by averaging over all nodes with responsibilities as weighting factors (see $\mathbf{t}_n \rightarrow \mathbf{x}(\mathbf{t}_n)$ on Figure 3.16):

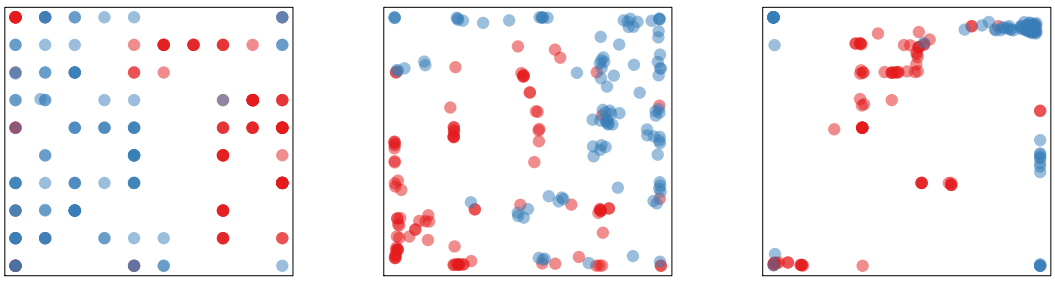
$$\mathbf{x}(\mathbf{t}_n) = \sum_{k=1}^{K} \mathbf{x}_k R_{kn} \tag{3.24}$$

Each point on the GTM map corresponds to the averaged position of one molecule. Responsibilities may be visualized on the 2D map for each molecule; we also used them for QSAR, to build property maps, and for database analysis.

3.5.2 Kernel GTM

KGTM is a variant of GTM introduced by Olier et al. [12], allowing the use of kernels as input data. As for KPCA, the preliminary step is to map the data into an implicit high-dimensional space using a kernel function; then, Olier. et al.'s algorithm is used the train a KGTM map. We implemented this kernel algorithm into our command-line

program GTMapTool, with different types of kernels (polynomial, linear, RBF, anova, etc.). In this algorithm, a new parameter appears: the dimensionality of the implicit feature space; it could be set as the intrinsic dimensionality of data, using a method like global or local PCA dimensionality estimation. Two examples of KGTM with an RBF and a polynomial kernel are shown in Figure 3.17; parameters were set arbitrarily to $[K = 100, M = 25, w = 1, \lambda = 1]$, the γ parameter regulating the RBF width was fixed at 0.2 and the dimension of the feature space at 20.



(a) Conventional GTM. **(b)** KGTM: RBF kernel. **(c)** KGTM: polynomial kernel.

Figure 3.17: (a) Conventional GTM, compared to kernel GTM maps with (a) an RBF kernel and (b) an polynomial kernel. The dataset contains inhibitors (red) and decoys (blue) of acetylcholinesterase, described by ISIDA fragment descriptors.

3.5.3 Incremental GTM

Bishop described in one article [11] the possibility of an incremental algorithm (iGTM). In the conventional GTM algorithm, the input data is considered as a single matrix of dimensions $N \times D$, where N is the number of molecules and D is the dimensionality of the input space, *i.e.*, the number of descriptors. When N is too large, it becomes impossible to keep this matrix in a computer's RAM. Unlike its conventional analogue, iGTM learns by small data blocks instead of using the whole data matrix at once. During the M-step, the parameters $\mathbf{W}$ and β are computed using both new and old responsibilities $\mathbf{R}'$ of N' molecules in a given data block $\mathbf{T}'$ corresponding to a data subset:

$$\mathbf{W} = (\mathbf{\Phi}^T \mathbf{G} \mathbf{\Phi} + \lambda \mathbf{I})^{-1} \left\{ \mathbf{R}\mathbf{T}^{\text{old}} + \left(\mathbf{R}'^{\text{new}} - \mathbf{R}'^{\text{old}} \right) \mathbf{T}' \right\} \tag{3.25}$$

$$\beta^{-1} = \beta^{-1} + \frac{1}{N'D} \sum_{n}^{N'} \sum_{k}^{K} \left(R'^{\text{new}}_{kn} - R'^{\text{old}}_{kn} \right) \left\| \mathbf{y}_k - \mathbf{t}'_n \right\|^2 \tag{3.26}$$

3.5.3.1 Implementation in GTMapTool

The following workflow was used in the command-line program developed during the thesis (GTMapTool):

1. The initial GTM manifold is initialized using the whole dataset with an incremental PCA, or using an initial random data subset, from which the GTM intrinsic parameters **W** and β are computed.

2. The first estimation of posterior probabilities (responsibilities) is performed incrementally (initialization E-step), using the **W** and β parameters computed in (1).

3. The first optimization of parameters is performed (initialization M-step), using the probabilities estimated in (2) .

4. Then, the actual incremental GTM starts, and the model is optimized by data blocks until convergence. Reaching convergence might require several dataset scans.

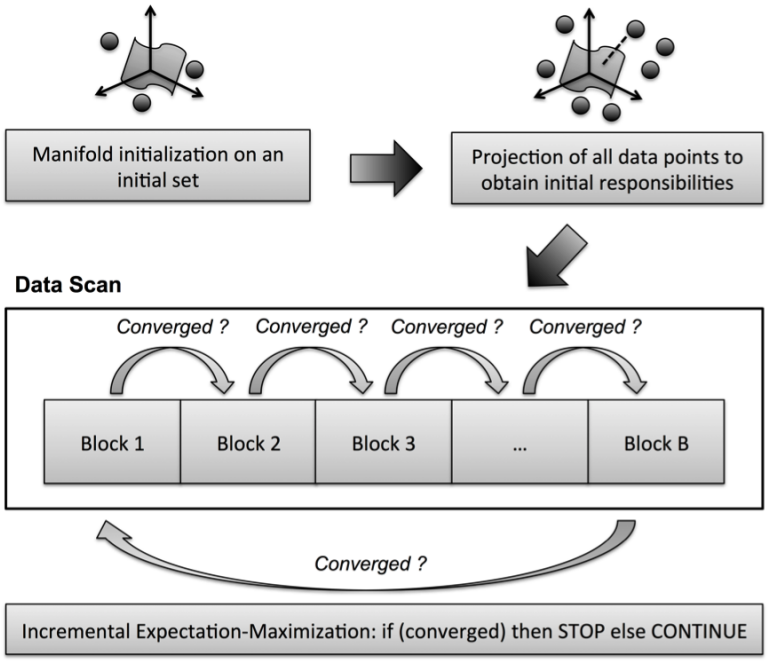


Figure 3.18: Incremental GTM process. An initial estimation of parameters and responsibilities is first performed; then, the model is optimized incrementally data block by data block. Several data scans (one scan = going through the whole dataset) might be necessary to reach convergence.

The whole process is illustrated in Figure 3.18. In this implementation, we use the "extended" floating point precision, but responsibilities are frozen if they reach the value

10E-100, to avoid computational instability. The number of blocks is a parameter that can be tuned by the user, just as the number of data scans (equivalent to the number of iterations in classical GTM). If the number of blocks is equal to the number of instances, then the model is optimized instance by instance; if it is equal to 1, it is equivalent to the classical GTM. There are two possibilities for checking iGTM convergence: block convergence and scan convergence.

For block convergence, the algorithm checks the log likelihood variation between two data blocks; it does not wait until it scanned the entire file. If the update is small enough (usually 10^{-4} log likelihood units), the algorithm stops. For scan convergence, the algorithm checks the log likelihood gain between two entire data scans. The block convergence, which usually gives the same result in a shorter amount of time, was chosen: if convergence is reached after a few number of data blocks, the algorithm stops without going through the rest of the dataset. The paper by Ng and McLachlan [65] gives simple rules for determining the number of blocks for faster IEM (incremental expectation-maximization) algorithms, which should be a trade-off between minimizing the number of M-steps and maximizing the speed of M-step computations.

3.6 InfoVis Techniques for High-Dimensional Data

3.6.1 Introduction

Information visualization (InfoVis) techniques give visual and/or interactive representations of data, and rely on the interpretation of users. The principle is to give new "points of view" on a dataset. These methods can be used on the original data, but also on data transformed by a dimensionality reduction algorithm. In this section, we give some examples of information visualization techniques which may help for visualizing high-dimensional data. In Table 3.4, some types of visualizations techniques useful for that purpose are listed. We mostly focused on table representations, the "parallel coordinates" family and iconographic displays, which are easily applicable to any dataset and are available in R or python packages.

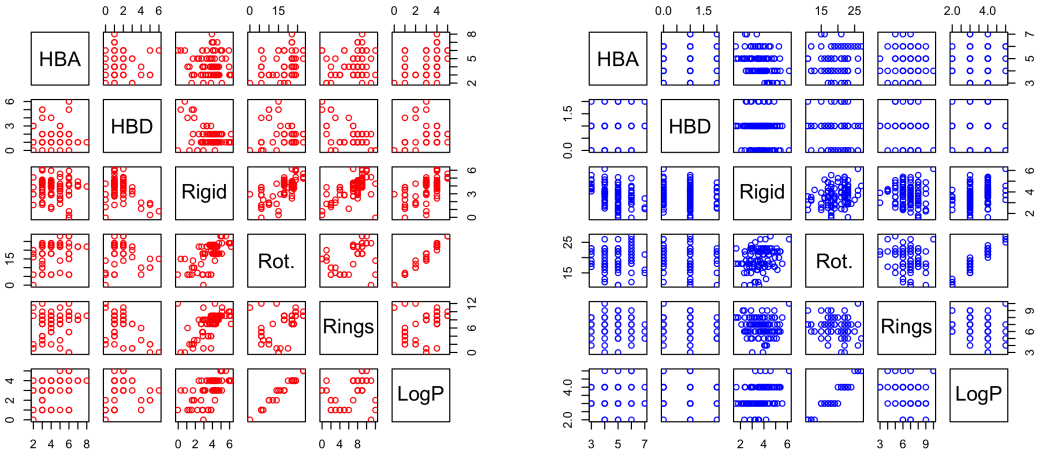
Table 3.4: Different types of visual structures.

Display Type	**Some examples**
Standard	scatterplots, bar plots
Tables	Bertin matrices, heatmaps
Parallel coordinates family	parallel coordinates, Andrews curves
Iconographic	Chernoff faces, star glyphs
Pixel-based	VisDB
Stacked displays	Tree maps, dimensional stacking

3.6.2 Some InfoVis Examples

3.6.2.1 Standard Approaches

The simplest visualization approach is a 2D (or 3D) scatterplot, where features (descriptors) can be visualized two by two (or 3 by 3). 2D or 3D scatterplots are also the commonest way of representing objects having gone through a dimensionality reduction algorithm, as we have shown in the section dedicated to these methods. Even with a high number of dimensions, a lot of people still want to look at the scatterplot matrices to detect simple patterns, which will sometimes not appear after using a complex machine learning model. An example of scatterplot matrix for 6 MOE descriptors describing 100 inhibitors and 100 decoys of acetylcholinesterase is given in Figure 3.19. These descriptors are the number of H bond acceptors O and N (HBA), the number of H bond donors OH and NH (HBD), the number of rigid and rotatable bonds, the number of rings, and logP (octanol/water partition coefficient).



(a) Ache inhibitors. **(b)** Ache decoys.

Figure 3.19: Scatterplot matrices for ache inhibitors (red) and decoys (blue), described by 6 MOE descriptors: number of H bond acceptors O and N (HBA), number of H bond donors OH and NH (HBD), number of rigid and rotatable bonds, number of rings, and logP (octanol/water partition coefficient)

3.6.2.2 Bertin's Permutation Matrices

Bertin's permutation matrices [66] (*matrices ordonnables*) are very simple data representations that can also be used as analysis tools. A data matrix is represented by a table; usually, for Bertin matrices, rows are features (descriptors) and columns in-

stances. Feature values are represented by bar graphs. After reordering rows in order to put similar features next to each other, the columns are similarly reordered so that clusters may appear. Heatmaps and heightmaps are variants of Bertin matrices, where descriptor values are represented by colors mapped on a 2D table or by heights on a 3D representation, respectively. An example of Bertin matrix in given in Figure 3.20, with descriptors re-arranged according to their correlation to data labels; the figure was generated with the *bertin* R package [67].

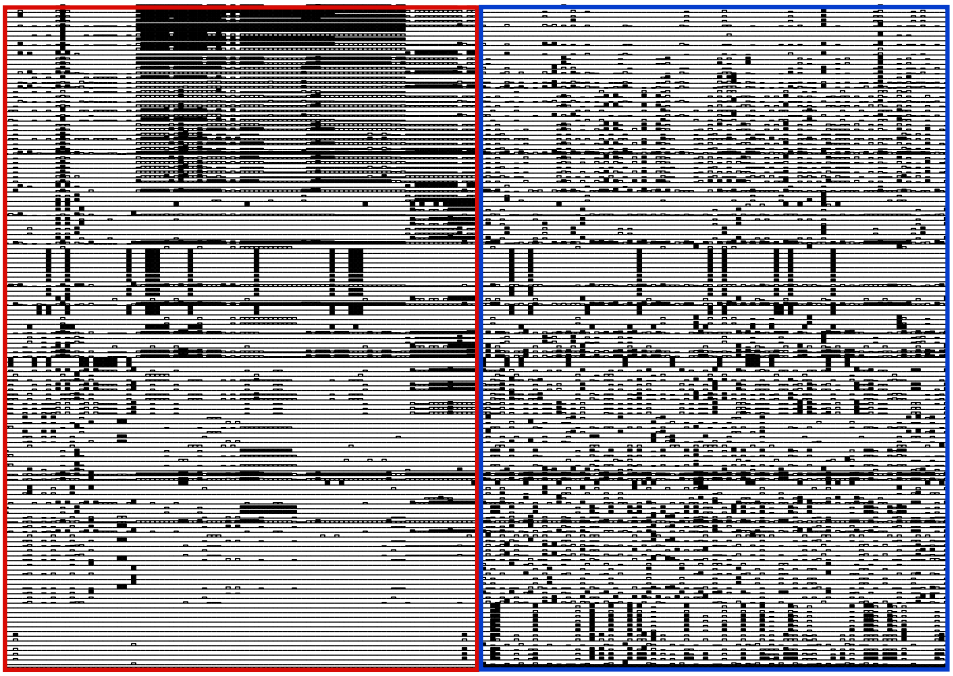
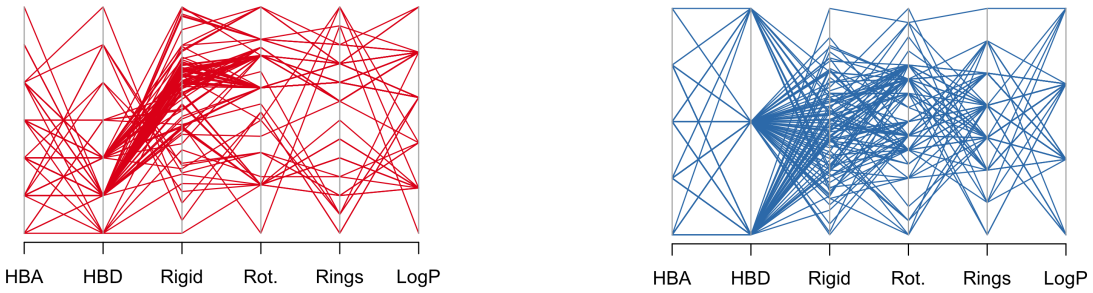


Figure 3.20: Bertin matrix, where rows are structural descriptors and columns 100 inhibitors (red) and 100 decoys (blue) of ache (acetylcholinesterase) described by ISIDA descriptors. The descriptors were re-organized depending on their correlation to the class (inhibitor/decoy).

3.6.2.3 Parallel Coordinates Family

Parallel coordinates [68, 69] are certainly the oldest and most popular visualization technique for datasets with a small number of dimensions. A line represents a data instance; the y-axis gives the value of the descriptor that can be found on the x-axis. In Figure 3.21, generated with the R package *MASS* [48], we show parallel coordinates for 100 inhibitors of acetylcholinesterase and 100 decoys, described by 6 MOE descriptors: number of H bond acceptors O and N (HBA), number of H bond donors OH and NH (HBD), number of rigid and rotatable bonds, number of rings, and logP (octanol/water partition coefficient). Variants of parallel coordinates include circular parallel coordi-

nates or RadViz [70] which considers springs attached to equally spaced "data anchors" on one end and to data points on the other, where the spring constant for each point and dimension is equal to the point coordinate. These methods share the disadvantage of being highly dependent on the ordering of the dimensions along the x-axis for parallel coordinates or around the circle for radial methods.



(a) Ache inhibitors. **(b)** Ache decoys.

Figure 3.21: Parallel coordinates for ache inhibitors (red) and decoys (blue), described by 6 MOE descriptors: number of H bond acceptors O and N (HBA), number of H bond donors OH and NH (HBD), number of rigid and rotatable bonds, number of rings, and logP (octanol/water partition coefficient).

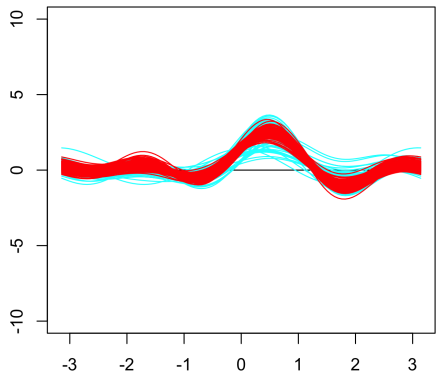


Figure 3.22: Andrews curves, where each curve represents one inhibitor (red) or decoy (blue) of acetylcholinesterase.

Andrews curves (1972) [71] are also an extension of parallel coordinates and provide a unique representation, which is not dependent on the order of descriptors. Instances (in our case, molecules) are represented by smooth curves on a 2-dimensional plot. Each data point $\mathbf{x}$ defines a Fourier series where coefficients are descriptor values $x_1, x_2, x_3, ...$ using the function $f(t)$:

$$f(t) = \frac{x_1}{\sqrt{2}} + x_2 sin(t) + x_3 cos(t) + ... \tag{3.27}$$

$f(t)$ can then be plotted from $t = -\pi$ to $t = \pi$. This method is especially useful for outlier detection, and is also used for clustering; an example is given featuring acetylcholinesterase inhibitors and decoys in Figure 3.22, for the same variables that we used for parallel coordinates in Figure 3.21. The figure was generated with the *andrews* R package [72].

3.6.2.4 Iconographic Displays

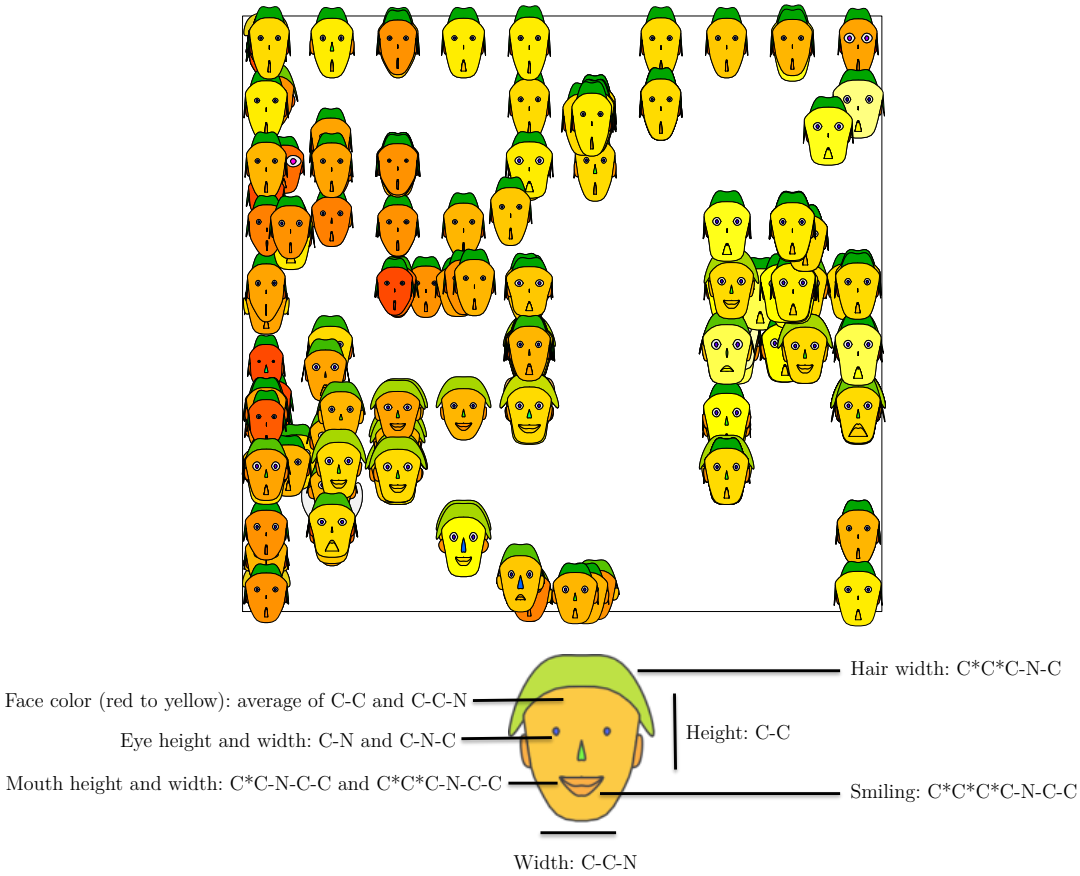


Figure 3.23: Application of Chernoff Faces to inhibitors and decoys of the acetylcholineesterase (one molecule = one face). The 2D coordinates were generated by GTM, and the face features were modified by structural descriptors; only a few of them are described here, where the symbol "*" represents an aromatic bond. This representation integrates dimensionality reduction information (2D GTM map) and original descriptors.

For iconographic techniques [73], each descriptor is encoded by a "glyph". These techniques are highly biased and the result is often difficult to interpret. Chernoff faces are certainly the most illustrative example of an iconic display. Two descriptors are used to build a scatterplot of "faces"; the other descriptors values are encoded by facial

features such as the nose, eyes, and so on. An example using the R package *aplpack* [74] is given in Figure 3.23; in this case, only 15 features can be encoded. For this Figure, we computed 2D coordinates with GTM for inhibitors and decoys of acetylcholinesterase from a 140-dimensional dataset, and used 15 structural descriptors to modify the faces (for example, the color of faces represents the number of CC and CCN fragments). An analogous representation is the star plot, where descriptors values are represented as spokes radiating from the center of a "star" glyph.

3.6.2.5 Hierarchical Techniques

Hierarchical techniques are most of the time used to deal with hierarchical data. Treemap [75] or Dimensional Stacking [76] are some examples; we could also include hierarchical clustering methods in this category. Treemap visualizes the data in the form of nested rectangles and is generally interactive; Dimensional Stacking was originally used for binary data, and divides a grid into embedded rectangles which represent dimensions.

3.6.2.6 Pixel-Based Methods

Pixel-based methods encode data values by pixels: a window, where each descriptor is mapped to a pixel colored by descriptor value, represents each data instance. An example of such system is VisBD [77].

3.6.3 Conclusion

All these visualization methods can be used in different circumstances. The Bertin matrix is certainly the most simple of the methods we have shown, and has several advantages: it gives a representation of the whole dataset without considerable modifications besides normalization, it may be used as a classification tool where important descriptors can be directly identified, and the visualization is made easy even for a high number of dimensions (we used 140 dimensions here). Parallel coordinates and related methods are only useful for a restricted number of descriptors (20), and they are dependent on the disposition of descriptors. However, by performing a PCA on the dataset and by ordering descriptors by decreasing eigenvalues, a fixed representation can be obtained. Andrews curves seem to be a self-sufficient visualization method, not dependent on the ordering of descriptors and useful for outlier detection. Finally, hierarchical techniques are useful for hierarchical data and pixel-based techniques for small databases visualization.

Chapter 4

Big Data Problem

In this chapter, we will discuss some technical issues related to the "big data" problem for GTM, which might arise when working with a high number of compounds, a high number of dimensions (*e.g.*, more than 10000), or both.

4.1 Technical Considerations

The original data matrix $\mathbf{T}$ is an $N \times D$ matrix, where N is the number of data points and D the number of dimensions. When we say, "this is a huge amount of data", it may mean different things: we may have a plethora of data points, a large amount of descriptors, or in the worst-case scenario (or the best, this depends from where you stand), an overabundance of both. The kernel version of GTM (KGTM) allows us to deal with an $N \times N$ matrix, which takes care of the dimensionality D problem, provided the number of data points N is not even bigger than D. The way to deal with the N issue is to use the incremental algorithm, which will process the data matrix data block by data block; but the incremental algorithm does not solve the D issue.

Therefore, if we have a very large data matrix both in terms of dimensionality and number of objects, we may find ourselves in a difficult situation. The kernel algorithm should then be avoided, and the incremental algorithm could be used but would take a long time to run. A combination of both kernel and incremental methods in a proper new algorithm could be a solution for this but we did not have the time to reflect upon this subject.

To deal with the D issue before running incremental GTM or iGTM, feature selection methods can be applied. A possible filter type will select descriptors with a given percentage of values different from 0, say 90%. Indeed, many sets of descriptors are in fact structural fragments or fingerprints (count or binary data types), which are very sparse, in the sense that many structural features are actually very rare in the dataset. However, it should be noticed that for some problems, the rare structural features are

actually the most interesting, and should not be removed. A lot of much more complex descriptor selection methods are available; the issue of feature selection has been briefly addressed in Chapter 3.

Finally, it should be noted that the GTM algorithm begins with a PCA initialization step for the initial positioning of the manifold; PCA has an $\mathcal{O}(D^2N + D^3)$ time complexity, $\mathcal{O}(D^2N)$ for the covariance matrix computation and $\mathcal{O}(D^3)$ for the SVD (singular value decomposition). The NIPALS algorithm for PCA [78] has an $\mathcal{O}(DNLI)$ complexity, where L is the number of selected principal components which should be 2 in our case and I is the number of iterations, which does not reach higher values than 5-10; therefore, the NIPALS algorithm should always be used when dealing with a high number of dimensions, since in our case L will be very small and equal to 2 in most cases. For incremental GTM, PCA must be performed incrementally (by computing the covariance matrix increment by increment) on the whole dataset, or by using an initial subset; both options were implemented into our command-line tool. Choosing this initial subset is an interesting problem, which is explored in further detail in the next section.

4.2 Initialize iGTM with a Small Subset

4.2.1 Introduction

The incremental version of the EM (expectation-maximization) algorithm of GTM may be convenient for decreasing the time needed to build a map for large datasets. However, the initialization step, *i.e.*, the initial positioning of the GTM manifold, still exists and may be time-consuming. One solution would be to select a subset of the data to initialize the manifold. However, the subset should be representative of the whole dataset. In this section, we try different selection techniques and evaluate the efficiency of a selected subset, before and after building the GTM. Here, we used a toy dataset of two intertwined rings each defined by 500 points in 3D to investigate the impact of the initial set. The intertwined rings dataset was downloaded from the Java SOMToolbox website belonging to the Vienna University of Technology [79].

4.2.2 Methods

For our iGTM maps (incremental GTM), we used a number of RBFs or radial basis functions $M = 25$, each of them with a width factor $w = 1$, $K = 625$ grid points, and a regularization coefficient $\lambda = 1$. The EM optimization stopped at convergence of the log likelihood function $\pm 10^{-4}$. We divided the dataset into 5 blocks of 200 instances, so that the iGTM (incremental GTM) would process the instances not all at once but by blocks of 200. All maps in this section were built using the iGTM algorithm. The GTM

Table 4.1: 9 different subsets used to initialize the iGTM model: 6 random subsets (r1, r2, r3, r4, r5, r6) and three diverse subsets selected with a diversity-based selection method (d1, d2, d3).

	Subset ID	Selection method	C1/C2
1	r1	random	3/7
2	r2	random	5/5
3	r3	random	3/7
4	r4	random	1/9
5	r5	random	3/7
6	r6	random	5/5
7	d1	MaxMin, random initial compound	5/5
8	d2	MaxMin, most representative initial compound	5/5
9	d3	MaxMin, most dissimilar initial compound	5/5

manifold was initialized with 9 different initialization sets comprising only 10 points from the dataset. The first 6 initialization sets were selected randomly. The three others were selected using the classical MaxMin DBCS (dissimilarity-based compound selection) algorithm [80] based on three steps:

1. Select an initial compound and place it in the subset.
2. Compute distances between instances in the dataset and instances in the subset.
3. Select the instance in the dataset with the highest distance to its closest neighbor in the subset and place it in the subset.
4. Return to step 2 until the required number of compounds in the subset is reached.

We used our own implementation of the algorithm, with Euclidean distances. The first instance to be put in the diverse subset (step 1 of MaxMin DBCS algorithm) was either chosen randomly, as the most similar ("most representative"), or as the most dissimilar to the other instances. All initial subsets are described in Table 4.1. A diversity score was established to assess the diversity of a specific subset, as the ratio of the average distance within the subset to the average distance within the entire dataset:

$$DiversityScore_{\text{subset}} = \frac{diversity_{\text{subset}}}{diversity_{\text{dataset}}}\% \tag{4.1}$$

4.2.3 Results and Discussion

In Figure 4.1, the initial manifold, the optimized manifold, and the data points of the two intertwined circles are represented. Nine GTM maps were computed with the same

whole dataset but using a different subset for positioning the initial manifold. It may be noticed that for subsets r2, d1 and d2 the initial manifold is positioned in the same plane as the one defined by one of the circles and cuts the other in two.

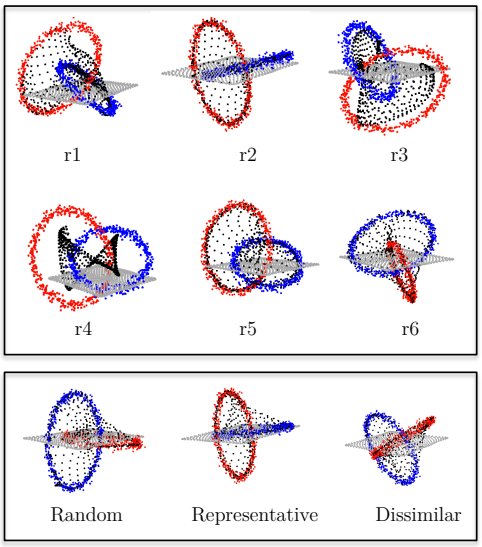


Figure 4.1: Initial manifolds (grey planes) and optimized manifolds (black points) fitted to the complete dataset (red and blue circles), built with GTM and initialized using 9 different initial subsets comprising 10 instances, (a) selected randomly, (b) selected with a MaxMin dissimilarity-based selection algorithm with a random (d1 subset), most representative (d2 subset) or most dissimilar (d3 subset) initial compound.

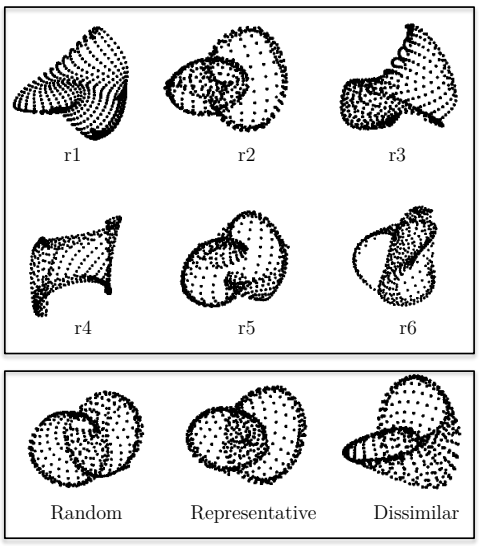


Figure 4.2: Optimized manifolds (black points) fitting the same dataset, but initialized using 9 different initial datasets containing 10 instances, (a) selected randomly, (b) selected using a MaxMin diversity-based selection algorithm with a random (d1 subset), most representative (d2 subset) or most dissimilar (d3 subset) initial compound.

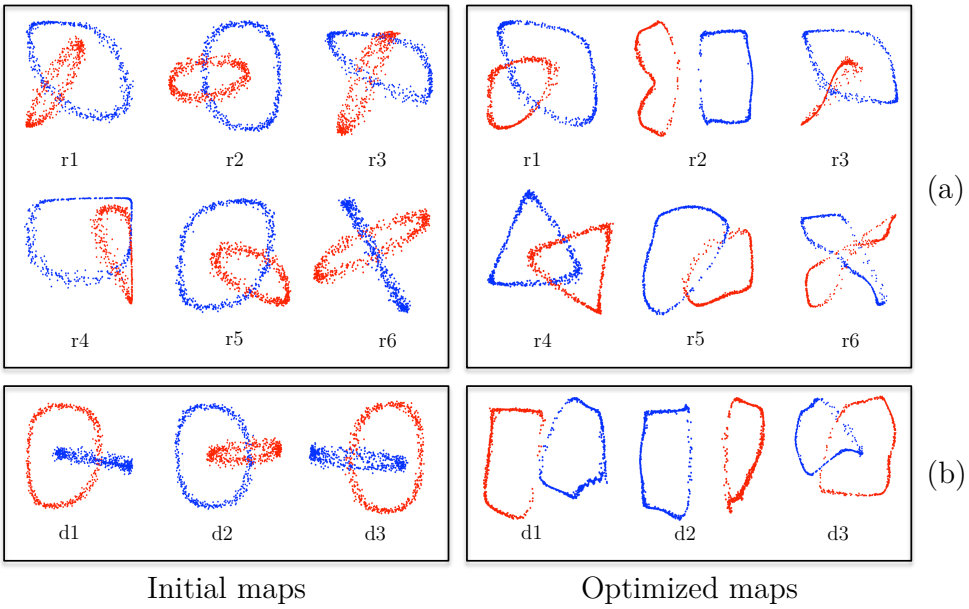


Figure 4.3: 2D GTM initial maps (left) and optimized maps (right) built using 9 different initial datasets containing 10 instances, (a) selected randomly, (b) selected using a MaxMin diversity-based selection algorithm with a random (d1 subset), most representative (d2 subset) or most dissimilar (d3 subset) initial compound.

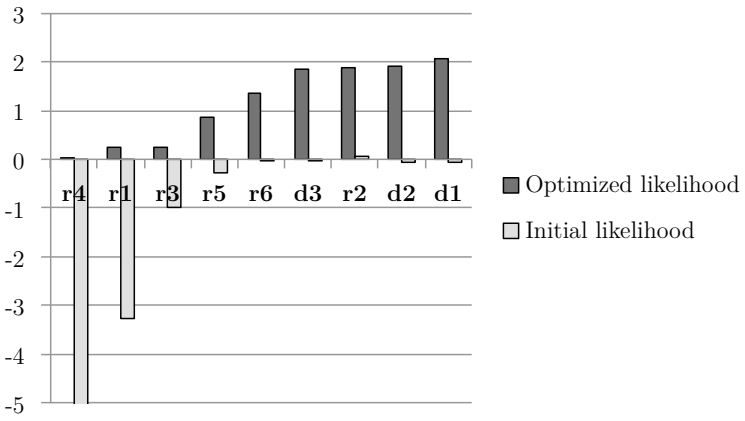


Figure 4.4: Initial and optimized log likelihood of 9 GTM maps built with the same dataset but initialized with 9 different subsets (see Table 4.1). The initial log likelihood of the GTM map initialized with r4 is out of range.

In Figure 4.2, only the optimized GTM manifold is represented. Manifolds initialized with r2, d1 and d2 can reconstruct the complete structure of both circles at the end of the optimization process. Manifolds initialized with d3, r5, and r6 subsets only form incomplete circles, and the worst cases correspond to initial subsets r1, r3 and r4, for which the original shape of the dataset is not recognizable. The worst manifold, with a

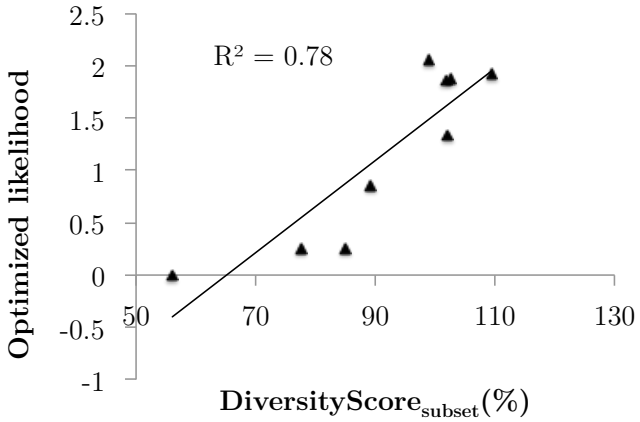


Figure 4.5: Optimized log likelihood of GTM maps initialized with specific subsets, as a function of subset diversity.

Table 4.2: Initial and final log likelihood, when using different initialization subsets with different sizes and proportions from each class (C1/C2); the mean μ and standard deviation σ of the log likelihood for the different subsets is also indicated.

	Nmol	C1/C2	Initial L	μ	σ	Final L	μ	σ
1	10	4/6	-0.159			0.94	1.168	0.384
2	10	3/7	-0.286	-0.152	0.137	0.95		
3	10	5/5	-0.011			1.61		
4	20	10/10	-0.054			0.98	1.004	0.094
5	20	13/7	-0.020	-0.045	0.022	1.11		
6	20	12/8	-0.061			0.92		
7	50	24/26	0.010			1.82	1.824	0.104
8	50	25/25	-0.020	-0.011	0.018	1.93		
9	50	24/26	-0.023			1.72		
10	100	42/58	-0.003			1.8	2.000	0.199
11	100	49/51	-0.007	-0.005	0.014	2.20		
12	100	53/47	0.020			2.05		
13	150	77/73	0.014			1.96	2.064	0.107
14	150	73/77	-0.006	0.009	0.014	2.17		
15	150	75/75	0.020			2.06		
16	200	95/105	0.010			2.23	2.262	0.05
17	200	98/102	0.001	0.008	0.007	2.23		
18	200	95/105	0.014			2.32		

rectangular shape twisted on the sides, was initialized with r4.

The corresponding 2D GTM maps are shown in Figure 4.3. In this figure, initial maps were obtained by projecting all the dataset onto the initial manifold built from

the subsets, and the optimized maps were obtained after optimization (they correspond to the manifolds shown in Figure 4.2). Only GTM maps initialized with subsets r2, d1 and d2 separated the two rings, and only after the optimization process. GTM maps initialized with d3 and r5 almost separated them, and the visually worst 2D map was initialized with r4. The manifolds initialized with r2, d1, and d2 were in the same plane as one of the rings, cutting the second in two parts (Figure 4.1), and, once optimized, matched the shape of the data (Figure 4.2); also, r2, d1, and d2 optimized maps were the only ones that separated the two intertwined rings (Figure 4.3, optimized maps).

The initial and optimized log likelihood of all 9 GTM maps were then considered and their values reported in Figure 4.4. The initial log likelihood corresponds to the state of the GTM at the initialization stage (initial manifold, built with the initial subset). The optimized log likelihood was obtained at convergence, after one or several scans of the entire dataset (optimized manifold). The three best GTMs according to the optimized log likelihood were initialized with r2, d2 and d1 and the worst with r3, r1 and r4, r4 being far below the others. Figure 4.4 also shows a link between the initial and optimized log likelihood. A GTM map that is not well initialized usually gives poor results (see r3, r1 and r4 in Figures 4.1, 4.2, 4.3). The $DiversityScore_{\text{subset}}$ was plotted against the optimized log likelihood in Figure 4.5. A link between the optimized log likelihood and the diversity score could be identified. This relationship between subset diversity and likelihood shows that the more diverse the initial subset is, the better the GTM model fits the data. We also investigated the impact of the initial subset size (randomly selected) on the log likelihood; results are given in Table 4.2. Results converge when reaching a subset size of 150-200 instances; this means that this map could be initialized with a random subset with a minimum size of approximately 150-200 instances.

4.2.4 Conclusion

The position of the initial manifold is important for fitting the data correctly with GTM. The selection of a diverse subset using a classical dissimilarity-based selection algorithm usually gives good results; however, it might be time-consuming for larger datasets, and this defeats the purpose of gaining time with an incremental version of GTM. We have shown that the more diverse the initialization subset is, the better the GTM visualization will be as well as the value of the log likelihood, *i.e.*, the GTM objective function. We have also shown that a good enough subset could be found randomly, and its quality for GTM initialization evaluated by the log likelihood function. We recommend that, for labeled data, the initial subset be representative of the population within each class; if the data is not labeled, representative samples can be selected using a stratified sampling scheme.

Chapter 5

Visualizing Descriptors or Properties

In this chapter, we introduce new methods for visualizing descriptors or properties on a 2D map. These methods provide a way to interpret a map obtained with GTM. The x-axis and y-axis of GTM maps my seem meaningless; they cannot be directly linked to any useful information. Visualizing original descriptors may be a way to understand a GTM model and draw more interesting conclusions; for example, a cluster will be characterized by specific descriptor values. We also show how to select regions of interest (ROIs), where several conditions are met to find objects (molecules) matching a profile.

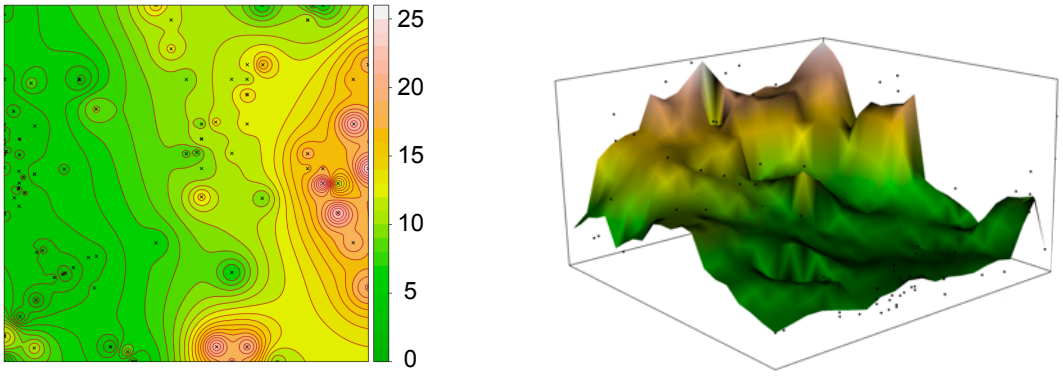
5.1 Activity or Descriptor Landscapes

GTM descriptor or activity landscapes are built by computing the average activity of training set molecules at each node $\mathbf{x}_k$:

$$\bar{a}_k = \frac{\sum_{n=1}^{N} R_{kn} \times a_n}{\sum_{n=1}^{N} R_{kn}} \tag{5.1}$$

where $\bar{\mathbf{a}}$ are the activity landscape values, $\mathbf{a}$ the activity values of the training set, R_{kn} the responsibility of the kth node for the nth molecule. These values can be used to generate a smooth 2D or 3D landscape using an interpolation method such as kriging [81, 82]. We used kriging in two different ways: by exploiting the previously calculated landscape vector $\bar{\mathbf{a}}$ associated with coordinates $\mathbf{X}$ (grid matrix), or by using training set experimental activities $\mathbf{a}$ at positions determined by the matrix $\mathbf{X}(\mathbf{T})$ containing the 2-dimensional coordinates of data points. Choosing the landscape activity vector $\bar{\mathbf{a}}$ with $\mathbf{X}$ is more accurate, since instead of using the data points' average positions we take all responsibilities $\mathbf{R}$ into account. Moreover, the kriging interpolation is computationally

very costly for big datasets and having a number of points limited to K grid nodes insures that the computation will not take too much time. For 2D and 3D landscapes



(a) 2-dimensional. **(b)** 3-dimensional.

Figure 5.1: 2D and 3D landscapes representing the stability of Lu(III)-ligand complexes.

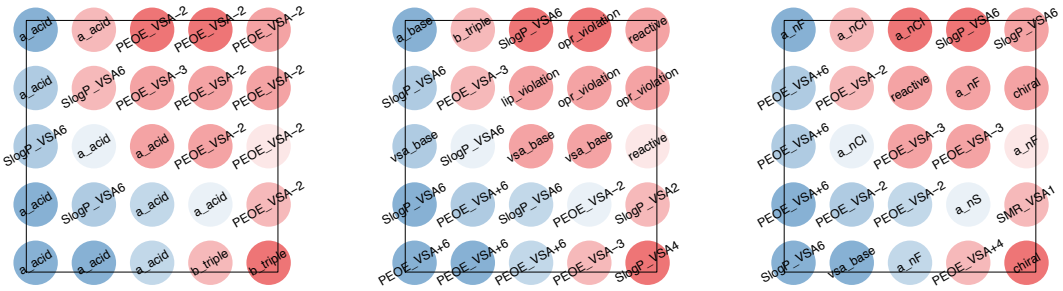
(Figure 5.1), the x-axis and y-axis delimit the GTM latent space; the activity $\bar{\mathbf{a}}$ is represented by colors in 2-dimensional landscapes, and by an additional z-axis in 3-dimensional landscapes. These landscapes can be built for descriptors or for properties.

The descriptor landscapes help to understand the organization of molecules on the 2D map. Regions of interest corresponding to certain descriptor criteria can be identified; for example, a region characterized by a high chirality and high molecular weight. We used this approach for the analysis of large chemical libraries by building 2D descriptor landscapes of 186 MOE 2D descriptors in Chapter 17.

The activity landscapes are tools for visualization and prediction; we used them as a basis for our GTM-based regression models in Chapter 16.

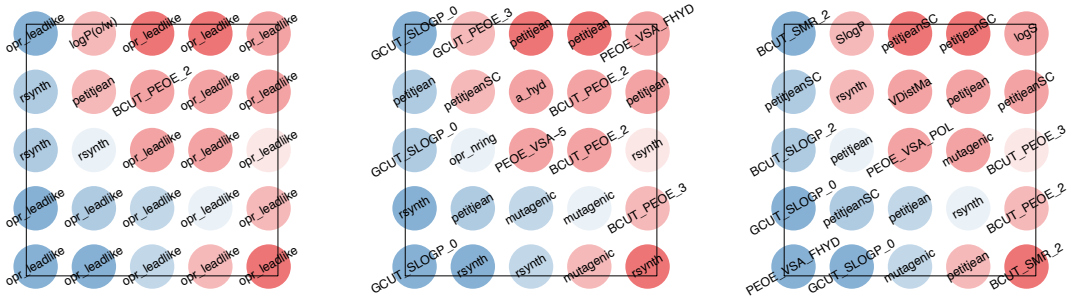
5.2 Descriptor Scores

In this section, we introduce three new descriptor scores to retrieve original descriptor information on GTM maps. The principle is to visualize on the 2D map the behavior of several descriptors instead of visualizing descriptor landscapes one by one. This analysis can be employed as a complement to descriptor landscapes. The three scores correspond to three questions we might ask about descriptors: at the kth node, which is the descriptor with the highest value? the descriptor with the lowest value? the descriptor for which molecules have the most similar values? The unscaled score matrices $\mathbf{S}(\max)$, $\mathbf{S}(\min)$ and $\mathbf{S}(\text{sim})$ with dimensions $K \times D$, where K is the number of nodes and D the number of descriptors, are calculated using the following equations, where $\mathbf{T}$ is the scaled data matrix with descriptor values lying in the $[0, 1]$ interval and



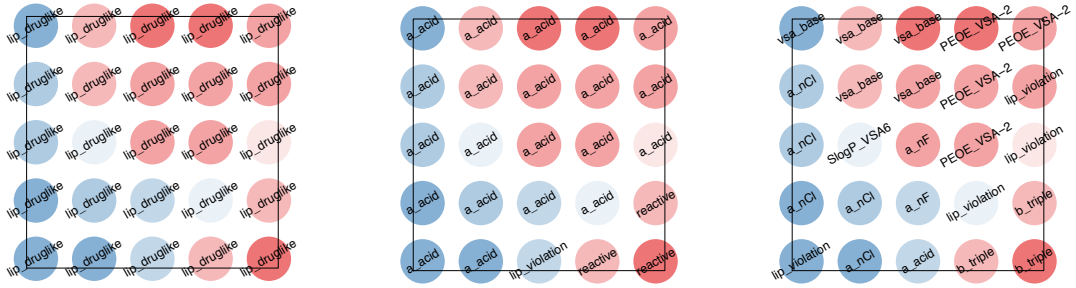
(a) First lowest values. **(b)** Second lowest values. **(c)** Third lowest values.

Figure 5.2: Descriptors with the lowest values on the map; the nodes are colored by the attributed class: red for inhibitors of ache and blue for decoys.



(a) First highest values. **(b)** Second highest values. **(c)** Third highest values.

Figure 5.3: Descriptors with the highest values on the map; the nodes are colored by the attributed class: red for inhibitors of ache and blue for decoys.



(a) First most similar values. **(b)** Second most similar values. **(c)** Third most similar values.

Figure 5.4: Descriptors with the most similar values; the nodes are colored by the attributed class: red for inhibitors of ache and blue for decoys.

R the responsibilities matrix:

$$S_{kd}(\max) = \sum_{n} R_{kn} \times T_{nd} \tag{5.2}$$

$$S_{kd}(\min) = \sum_n R_{kn} \times (1 - T_{nd}) \tag{5.3}$$

$$S_{kd}(\text{sim}) = \sum_n \sum_{n'} |R_{kn} \times T_{nd} - R_{kn'} \times T_{n'd}| \tag{5.4}$$

The final scores **Score**(max), **Score**(min) and **Score**(sim) are obtained after scaling as following:

$$Score_{kd} = \frac{S_{kd} - \min(\mathbf{s}_k)}{\max(\mathbf{s}_k) - \min(\mathbf{s}_k)} \tag{5.5}$$

These scores can be used to rank the descriptors according to different criteria: high value, low value, similar value at each node $\mathbf{x}_k$. In Figures 5.3, 5.2 and 5.4, we show a classification map of acetylcholinesterase inhibitors (red) v.s. decoys (blue) with a very low resolution (complete set of random parameters $[K = 25, M = 25, w = 10, \lambda = 100]$) to better illustrate the principle of descriptor scores. Each node is colored according to the predicted class, and the first, second and third best descriptor according to the score is indicated over each node: **Score**(max) in Figure 5.3 for descriptors with the highest values, **Score**(min) in Figure 5.2 for descriptors with the lowest values, and **Score**(sim) in Figure 5.4 for descriptors with the most similar values. For example, in

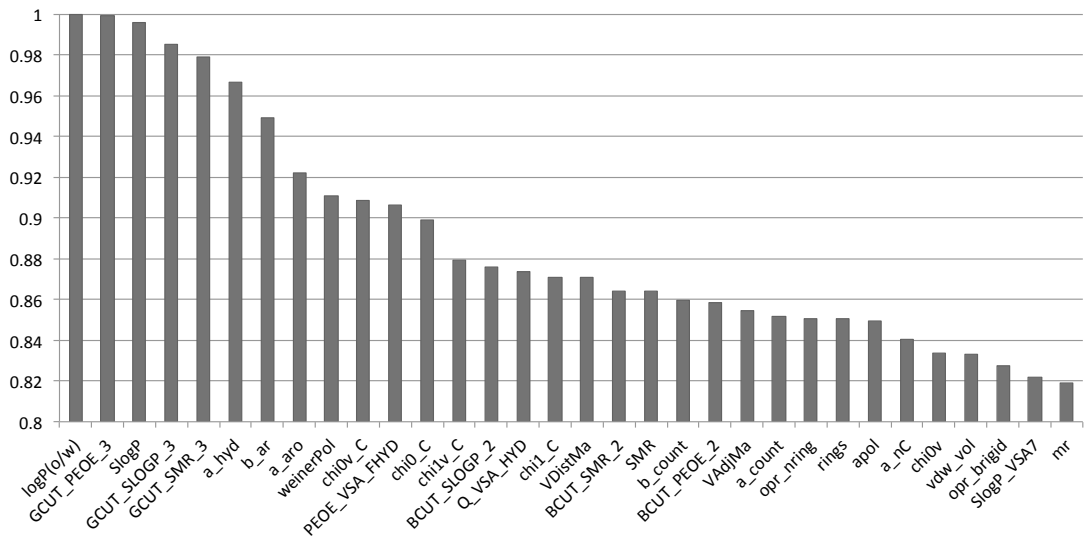
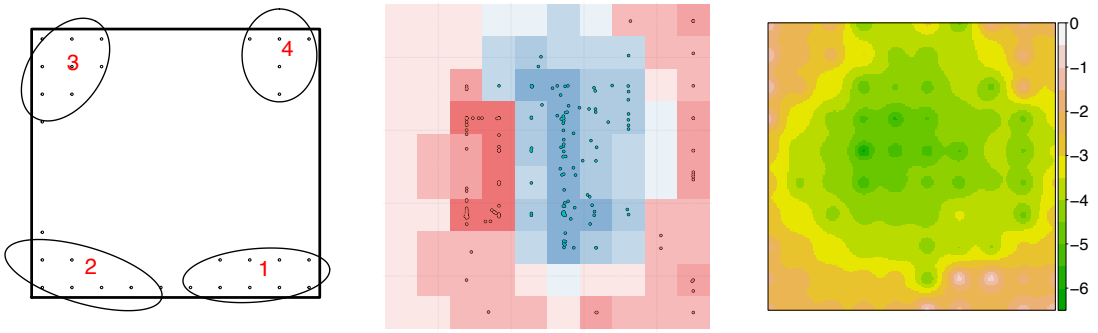


Figure 5.5: Descriptors with highest values at a given node of the GTM map, with $Score(\max) > 0.8$.

Figure 5.3 one node dominated by inhibitors (on the top row, the second from the left) is characterized by a lower opr_leadlike score than most other nodes, and a high logP and SlogP value, which means that we might find hydrophobic inhibitors of ache in this region. If we are interested in this particular node, the complete score vector of a node

score$_k$ can be plotted in a bar plot, as shown in Figure 5.5, showing descriptors with the highest values at this node (with score > 0.8).

5.3 Regions of Interest (ROIs)



(a) Four regions of interest. **(b)** Ache classification map. **(c)** LogS landscape.

Figure 5.6: (a) Four regions of interest (ROIs) corresponding to (b) dominance of inhibitors of acetylcholinesterase (red) over decoys (blue) on the classification map and (c) to high solubility (>-2.5) on the LogS landscape.

If we are interested in several descriptors or several properties, we can superimpose landscapes to find regions of interest (or ROIs) corresponding to several criteria. An ROI example in a chemical space of protein ligands could be a region of high solubility, low chirality and high affinity for a target. This method could be used to find regions corresponding to an entire activity profile.

A set of landscape vectors is first generated: $\{\bar{\mathbf{a}}_1, \bar{\mathbf{a}}_2, \bar{\mathbf{a}}_3, ...\}$. Then, ROIs are found by applying a condition to each landscape node; ROIs correspond to nodes where all these conditions are met. Molecules in ROIs can be retrieved automatically. The simplest approach is to draw an ellipse around each node or group of nodes and retrieve compounds within the ellipse. If the groups of nodes are easily identified, a k-means procedure or any clustering approach can be used to find the nodes' clusters and a confidence ellipse can be drawn around the region. If groups of nodes cannot be easily identified, the best approach is to retrieve compounds node by node. In Figure 5.6, four regions exhibiting a "high" solubility (logS $>$ -2.5) and a dominance of ache inhibitors (red class) were drawn on a GTM map; the parameters used in this case were $[K = 100, M = 25, w = 1, \lambda = 1]$.

Chapter 6

Optimization of models

6.1 Training and Test

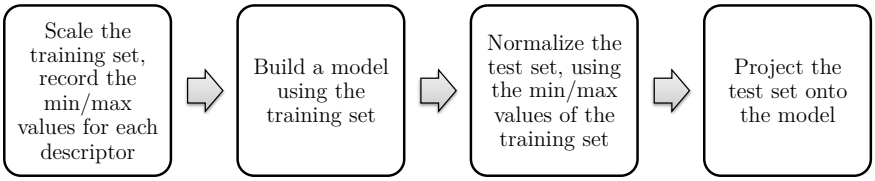


Figure 6.1: GTM training and test sets normalization workflow.

Our goal is to build a model able to predict characteristics of new compounds; typically, their localization in a cluster (clustering task), the class they belong to (classification task), or their activity (regression task). To build a model, we need a training set, which contains compounds we want to learn from. For many machine learning methods, the training set descriptors have to be normalized, so that each descriptor value lies in the same interval; this can be done by max/min normalization (eq. 5.1) or by standardization (eq. 5.2) of the original data matrix $\mathbf{O}$ to obtain the normalized matrix $\mathbf{T}$; we used max/min normalization in all our calculations to obtain values ranging from 0 to 1:

$$T_{nd} = \frac{O_{nd} - \min(\mathbf{o}^d)}{\max(\mathbf{o}^d) - \min(\mathbf{o}^d)} \tag{6.1}$$

$$T_{nd} = \frac{O_{nd} - \sigma(\mathbf{o}^d)}{\mu(\mathbf{o}^d)} \tag{6.2}$$

The maximum and minimum values of each descriptor $\mathbf{o}^d$ are stored and used to scale the test set containing new, unseen compounds. For GTM models, the test set is projected onto the manifold built with the training set.

6.2 Model Validation Procedure

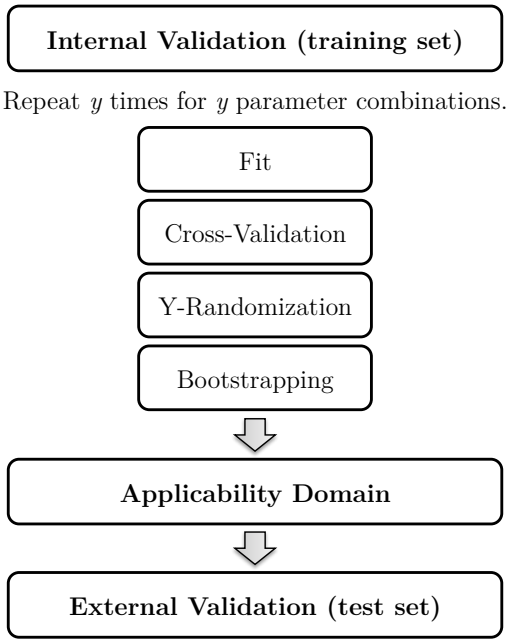


Figure 6.2: Model validation workflow.

The classical QSAR validation procedure in chemoinformatics involves dividing the dataset into training and test sets. The training set is used for the internal validation. The internal validation process consists most of the time in cross-validation [83] (CV), a procedure to find the best parameters according to a CV performance score. There are other internal validation techniques, such as fit, y-randomization [84] or bootstrapping [85]. After selecting the best model using CV and building it, an applicability domain (AD) can be established to define the limitations of the model. Using y-randomization together with CV for internal validation is a good policy to evaluate the robustness of the model.

The external test set can be used to assess the prediction error (external validation), and is used only once; the model should never be "aware" of it before the external validation step.

It has been argued by Baumann et al. [86] that a better validation process would involve a double cross-validation, where an inner loop would be dedicated to internal validation (model selection) and an outer loop to external validation (model assessment). However, in our applications to chemical datasets, we mainly used the conventional internal and external validation procedures to avoid computational costs.

For the sake of completeness, we summarize the most common methods [87] used to perform the internal validation of a model:

Fitting consists in building a model from the entire training set given a specific set of parameters and descriptors, and evaluating the performance of the model based on its ability to predict the training set instances, *i.e.*, the same instances that were used to build the model.

Cross-validation [83] or CV consists in dividing the training set into n folds; a model is trained with $n-1$ folds, and evaluated using the remaining fold; the procedure is repeated n times, each time changing the fold used for testing, so that all folds at the end have been tested once.

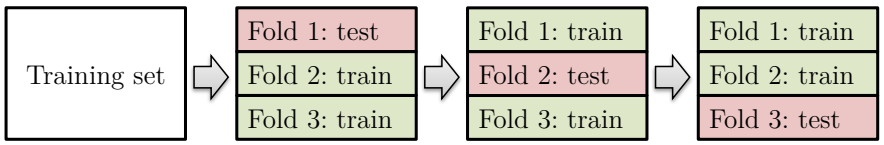


Figure 6.3: Cross-validation procedure: a training set is divided into n folds (here $n = 3$); n models are then built, each trained with $n - 1$ folds, and evaluated using the remaining fold.

Bootstrapping [85] comes from the expression "to pull oneself up by one's bootstraps", which means to accomplish an impossible task. This method is often used when performing induction from very small datasets, and is based on sampling with replacement; "with replacement" means that after picking randomly one object from a bag containing all our data, we place it back in the bag (we "replace" it), so it can be picked again next time. The procedure involves selecting n groups of objects by sampling with replacement from the available data. For each of the n training groups, n models are built and applied to test compounds which were not included in the training group. This method is much more tedious than cross-validation.

Y-Randomization [84] or scrambling can be useful to establish whether or not our performances were due to chance, by randomizing the y variable (the dependent variable). Thus, this methods relies on the presence of a dependent variable, such as classes or properties to be predicted; the y values must be assigned randomly to the objects in the dataset. The randomization/prediction process is repeated many times, and the performances of models are compared to those without randomization. If the results are too close, the models are deemed inefficient. This procedure can be coupled with cross-validation; for example, comparing the outcome of normal v.s. y-randomized cross-validation.

6.3 GTM Parameters

Four parameters should be optimized in the GTM algorithm: the number of RBFs (radial basis functions) M, the RBF width factor w, the number of grid nodes K and the regularization coefficient λ. K and M define two regular grids in the 2-dimensional

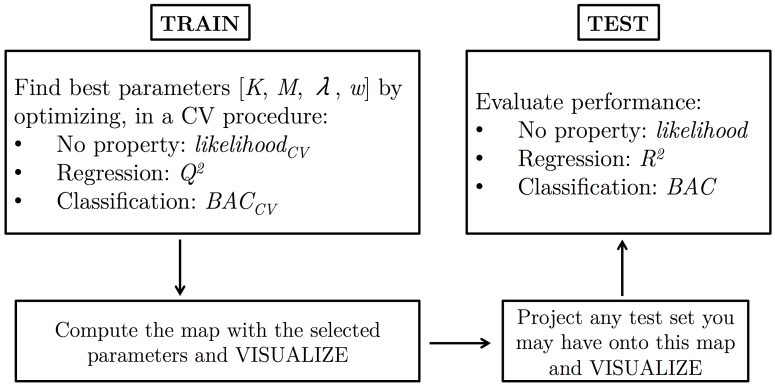


Figure 6.4: The four GTM parameters can be optimized by internal cross-validation on the training test and external validation on the test set, using the log likelihood or performance measures such as BAC (balanced accuracy) and R^2 (determination coefficient).

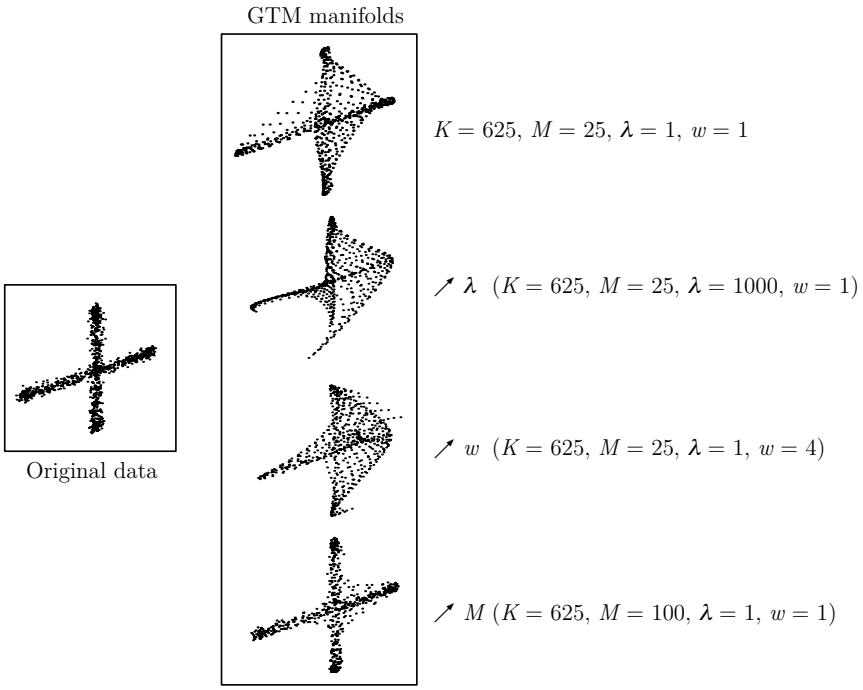


Figure 6.5: Original 3D plot of two intertwined rings (side view), and 4 plots of manifolds trained on the original data with different parameters. This figure shows how the shape of the manifold can be changed by tuning the M, w and λ parameters.

space: the grid of nodes and the grid of RBF centers. K may be seen as a resolution parameter; beyond a certain value (we do not use more than 1600 nodes generally, corresponding to a 40 by 40 grid), it will not improve the performance of the model but only cost more computation time; in any case, it should not be greater than the number of molecules. M also increases computation time, but we noticed that using large M values (equal to K for example) could greatly increase the performance of a

model. With a fixed K parameter, the shape of the manifold can be tuned with the M, w and λ parameters; too small w and λ parameters or high M might generate a too complex model, that will shape the training data too well, and induce overfitting (cf. Figure 6.5). We generally vary λ from 10^{-n} to 10^{n} where n takes values from -4 to 3, w from 0.25 to 2 or 3, M from 2×2 to 25×25 and K from 10×10 to 50×50. The four GTM parameters can be optimized by monitoring the log likelihood function, if no target activity is available; for QSAR (quantitative structure-activity relationship), it is the predictive ability of the model that should be optimized (cf. Figure 6.4). The unsupervised and supervised optimization procedures are detailed in the following sections.

6.4 GTM Unsupervised Optimization

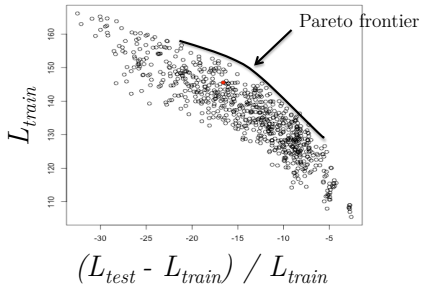


Figure 6.6: The model can be optimized by choosing a trade-off between the average train log likelihood in CV and the difference between the average test and train CV likelihoods.

The GTM unsupervised optimization is based on the log likelihood, the GTM objective function. A log likelihood value $\mathcal{L}_n(\mathbf{W}, \beta)$ can be computed for each molecule (instance) $\mathbf{t}_n$:

$$\mathcal{L}_n(\mathbf{W}, \beta) = ln \left\{ \frac{1}{K} \sum_k p(\mathbf{t}_n | \mathbf{x}_k, \mathbf{W}, \beta) \right\} \tag{6.3}$$

In this thesis, we always used an average log likelihood value $\mathcal{L}_{norm}(\mathbf{W}, \beta)$ to compare different datasets:

$$\mathcal{L}_{\text{norm}}(\mathbf{W}, \beta) = \frac{\sum_n \mathcal{L}_n(\mathbf{W}, \beta)}{N} \tag{6.4}$$

This formula can be used to compute the average train and test log likelihoods L_{train} and L_{test} obtained in CV:

$$L_{\text{train}} = \frac{\sum_{n=1}^{n_{\text{CV}}} \mathcal{L}_{\text{norm}}(\text{train}_i)}{n_{\text{CV}}} \tag{6.5a}$$

$$L_{\text{test}} = \frac{\sum_{n=1}^{n_{\text{CV}}} \mathcal{L}_{\text{norm}}(\text{test}_i)}{n_{\text{CV}}} \tag{6.5b}$$

where $\mathcal{L}_{\text{norm}}(\text{train}_i)$ and $\mathcal{L}_{\text{norm}}(\text{test}_i)$ are the normalized log likelihoods of the train and test fold i, respectively, and n_{CV} the number of folds. The goal is to obtain a large log likelihood; however, a train log likelihood much larger then a test log likelihood may indicate that the model overfits the training set. A Pareto efficiency plot [88, 89] (see Figure 6.6) can be used to choose a good trade-off between the train log likelihood and the difference between train and test log likelihoods.

6.5 GTM Supervized Optimization

In this thesis, we often used a type of supervized optimization during the internal validation step, by choosing the model according to the prediction performance instead of the values of train and test log likelihoods. For regression, we used the root-mean-square error as a measure of accuracy:

$$RMSE_{\text{CV}} = \sqrt{\frac{\sum_n (a(\text{pred})_i - a(\text{exp})_i)^2}{N}} \tag{6.6}$$

where $a(\text{pred})_i$ is the predicted activity of the ith compound during cross-validation (compound in the "test fold" in Figure 6.3), $a(\text{exp})_i$ the experimental activity, and N the total number compounds exploited for cross-validation. We also used the cross-validated determination coefficient [90, 91] Q^2:

$$Q^2 = 1 - \frac{\sum_n (a(\text{exp})_i - a(\text{pred})_i)^2}{\sum_n (a(\text{exp})_i - \bar{a}(\text{exp}))^2} \tag{6.7}$$

where $\bar{a}$ is the average experimental activity; Q^2 is less than or equal to one and may have negative values if the predictions are worse than those obtained with a model always predicting the sample mean. The formulas for external validation indicators $RMSE$ and determination coefficient R^2 are the same as for cross-validation indicators $RMSE_{\text{CV}}$ and Q^2; however, for external validation, the predicted and experimental activities $\mathbf{a}(\text{pred})$ and $\mathbf{a}(\text{exp})$ are assigned to a single test set predicted with a single model. Models with a Q^2 below 0.5 should be discarded; they should also achieve at least $R^2 = 0.6$ in external validation [92]. For classification tasks, we used several performance measures, but usually opted for the balanced accuracy (BAC) [93]:

$$BAC = \frac{1}{N_c} \sum_n \frac{TP_i}{TP_i + FN_i} \tag{6.8a}$$

$$BAC = \frac{1}{N_c} \sum_i recall_i \tag{6.8b}$$

where N_c is the number of classes, TP_i the number of true positives for the ith class (instances correcly predicted as belonging to class c_i) and FN_i the number of false negatives for the ith class (instances incorrecly predicted as not belonging to class c_i). The sum of TP and FN equals P, the total number of instances actually belonging to the class. The balanced accuracy can be seen as an average of recalls for the different classes. The recall is actually the proportion of relevant instances that the model was able to retrieve. It answers the question: "What percentage of instances actually belonging to the class have I been able to identify?":

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{6.9}$$

Another useful indicator is the precision, which measures the proportion of retrieved instances that actually belong to the class; the question is then: "From the instances that I identified (correctly or not) as belonging to the class, how many actually do belong to the class?":

$$precision = \frac{TP}{TP + FP} \tag{6.10}$$

The $F - score$, the harmonic mean of recall and precision, can be used was well as the BAC for the optimization of GTM models:

$$F - score = 2\frac{recall \times precision}{recall + precision} \tag{6.11}$$

Chapter 7

GTM Classification Models

GTM can be applied to classification tasks using its probability density functions (PDFs) in the initial space; this was already shown in our 2012 paper [1]. In this chapter, we also introduce the classification in latent space using posterior probabilities or responsibilities [2], and establish protocols to build GTM-based 2D classification models and visualize them.

7.1 Initial Space Classification

To perform classification tasks, there are two possibilities: either fit one manifold per class or use the same manifold for all classes. Our first initial space models [1] were designed to fit one manifold per class, generating one data density $p(\mathbf{t}|c_i)$ for each class c_i. The class probability $P(c_i|\mathbf{t})$ can be obtained by applying Bayes' theorem:

$$P(c_i|\mathbf{t}) = \frac{p(\mathbf{t}|c_i) \times P(c_i)}{\sum_i p(\mathbf{t}|c_i) \times P(c_i)} \tag{7.1}$$

where $p(\mathbf{t}|c_i)$ is the approximation of the density generated in the data space:

$$p(\mathbf{t}|c_i) = \frac{1}{K} \times \sum_k p(\mathbf{t}|c_i, \mathbf{x}_k) \tag{7.2}$$

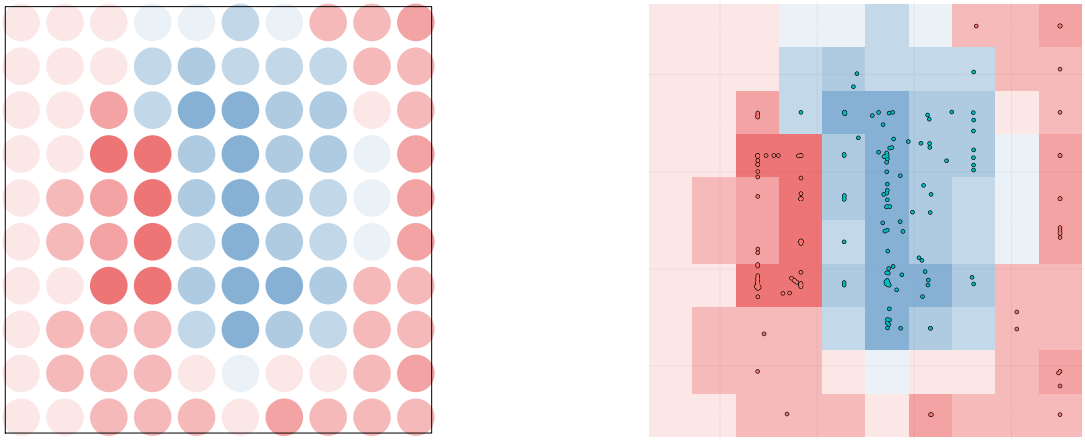
and $P(c_i)$ is the prior for class c_i:

$$P(c_i) = \frac{N_{c_i}}{N} \tag{7.3}$$

N_{c_i} being the number of molecules in class c_i and N the total number of molecules. This methodology was applied to the classification of actives and decoys from the DUD (directory of useful decoys), in our article shown in Chapter 14. The class of new molecules can be predicted without building the model once again: the molecules are

projected onto both manifolds, in order to obtain new data densities $p(\mathbf{t}_q|c_i)$ for each qth new molecule given a class c_i. Then, the above formula can be used to predict their activity profile $P(\mathbf{c}|\mathbf{t}_q)$. However, the initial space classification model cannot be visualized; this is why we introduced the latent space classification.

7.2 Latent Space Classification



(a) Node representation. **(b)** Grid representation.

Figure 7.1: Two visual representations of GTM classification model with applicability domain (AD), for ache inhibitors (red) and decoys (blue). Lighter regions have a lower probability given the winning class $P(\mathbf{x}_k|c_{\max})$ and are outside the AD. Points in map (b) represent individual compounds colored by class.

Instead of using the data density in initial space, we may use the posterior probabilities or responsibilities (cf. Chapter 3). Thus, instead of computing the probability of a class given the data, we compute the probability of a class given a node, using Bayes' theorem once again:

$$P(c_i|\mathbf{x}_k) = \frac{P(\mathbf{x}_k|c_i) \times P(c_i)}{\sum_i P(\mathbf{x}_k|c_i) \times P(c_i)} \tag{7.4}$$

where $P(\mathbf{x}_k|c_i)$ is computed as follows:

$$P(\mathbf{x}_k|c_i) = \frac{\sum_{n_i}^{N_{c_i}} R_{kn_i}}{N_{c_i}} \tag{7.5}$$

where R_{kn_i} is the responsibility of node $\mathbf{x}_k$ for a molecule belonging to class c_i. Using these formulas, two methods can be used to classify new query compounds.

1. **Global method:** to predict the activity profile $P(\mathbf{c}|\mathbf{t}_q)$ of the qth query com-

pound with associated responsibilities $\{R_{kq}\}$, the following equation can be used:

$$P(c_i|\mathbf{t}_q) = \sum_k P(c_i|\mathbf{x}_k) \times R_{kq} \tag{7.6}$$

We named this method the "global" prediction, since it employs all responsibilities of a query $\mathbf{t}_q$ to generate the activity profile.

2. Local method: The "local" prediction only uses the conditional probability of the node closest to the molecule in 2D $P(\mathbf{x}_{\text{nearest}}|c_i)$:

$$P(c_i|\mathbf{t}_q) = P(\mathbf{x}_{\text{nearest}}|c_i) \tag{7.7}$$

The winning class will have the highest $P(c_i|\mathbf{t}_q)$ value.

The main advantage of using latent space classification models is the associated visualization. Each GTM node will have a probability value $P(\mathbf{x}_k|c_{\max})$, where c_{max} is the winning class. Nodes can be represented as circles colored by class. The probability $P(\mathbf{x}_k|c_{\max})$ assigned to each node can be used to change their shape or transparency. We give in Figure 7.1 two possible representations of $P(\mathbf{x}_k|c_{\max})$ by transparency variation, using the ache dataset from the DUD database.

Chapter 8

GTM Regression Models

Regression consists in learning a hidden relationship between explanatory variables x and dependent variables y from several observations. For QSAR (quantitative structure-activity relationship) tasks, dependent variables are activities to be predicted, such as the affinity for a given target or the solubility of molecular compounds. We designed new GTM-based supervised and unsupervised regression methods; supervised approaches use activity information as well as descriptors to train the models, whereas unsupervised methods only use descriptors and are "blind" to the activities during the training process.

8.1 Unsupervised GTM Regression

Our models based on conventional GTM are built by fitting a manifold to a training set in descriptor space. Therefore, they are blind to the activities of the training set, which are only used after building the model. We designed several regression methodologies, consisting of a "global" method based on the whole probability distribution of a query compound $\mathbf{t}_q$ on the 2D GTM map, and "local" methods taking into account only the neighborhood of $\mathbf{t}_q$.

1. Global Method. The global prediction approach is based on the K-dimensional activity landscape vector $\bar{\mathbf{a}}$ computed as described in Chapter 5, with responsibilities of a GTM based on a specific descriptor space, and some training set activities. The set used for building the GTM map and the one providing the experimental activities do not have to be the same. The GTM map can be built with a first set, and a second set of compounds can be projected on the map to obtain new responsibilities, used together with their activities to build a landscape. The global method uses the responsibility R_{kq} of each node $\mathbf{x}_k$ for a query molecule $\mathbf{t}_q$ as a weight for the activity landscape value $\bar{a}_k$ at node $\mathbf{x}_k$, so that the predicted activities $\hat{\mathbf{a}}$ of queries are weighted averages of

landscape values:

$$\hat{a}_q = \sum_{k=1}^{K} R_{kq} \times \bar{a}_k \tag{8.1}$$

2. Nearest nodes. The nearest nodes approach consists in selecting the V nearest nodes $\mathbf{x}_v$ of molecules on the 2D GTM map using Euclidean distances, and retrieving their activity landscape values $\bar{a}_v$; Euclidean distances are used to build the GTM probability density functions (PDFs) during training and should be a natural choice for this task. The predicted activity of a molecule query is the average of landscape activity values of these V nearest nodes:

$$\hat{a}_q = \frac{\sum_v^V \bar{a}_v}{V} \tag{8.2}$$

3. Nearest neighbors. The nearest neighbors method (classical k-NN) is the same as the nearest nodes approach, except that nearest training set compounds are retrieved instead of nearest nodes. These nearest neighbors can be found by computing Euclidean distances between the mean position of compounds on the 2D map or by estimating the correlation between their responsibilities; the 2D approach is usually more performant (see Chapter 16). The predicted activity of a new tested compound $\mathbf{t}_q$ is the average of experimental activities $\{a_v\}$ of the V nearest training set compounds:

$$\hat{a}_q = \frac{\sum_v^V a_v}{V} \tag{8.3}$$

All these GTM-based unsupervised prediction approaches were tested on several datasets and compared to classical machine learning methods in Chapter 16.

8.2 Supervised GTM Regression with S-GTM

We developed regression methods to predict individual activities or whole activity profiles with our new Stargate GTM or S-GTM method. With S-GTM, molecules can travel from an A-dimensional descriptor space to a B-dimensional activity space. This mapping from one space into another generates coordinates in the activity space. Models are trained using descriptors and activities; therefore, the method is supervised. Details about the algorithm and prediction methodology are given in Chapter 11.

Chapter 9

GTM Applicability Domain

An applicability domain (AD) in chemoinformatics defines the domain in chemical space where a model's prediction is applicable; it is dependent on the instances used to train the model as well as on the features used to model the space. We introduce in this chapter some new AD definitions for GTM models.

9.1 Applicability Domain and Outliers

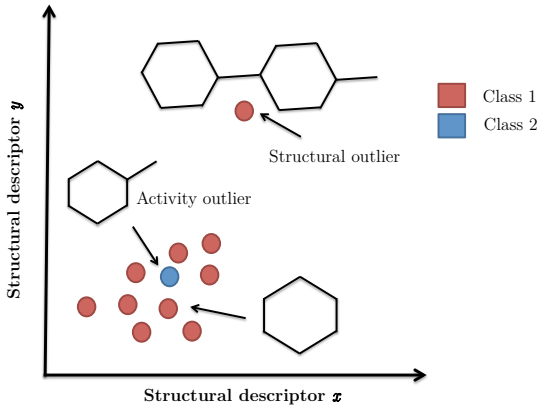


Figure 9.1: There are two main types of outliers: activity and structural outliers. Different types of applicability domains may reject these molecules that cannot be well described by a given model.

Applicability domains (ADs) and outliers are two closely related concepts. A model prediction for a new compound will not be trusted if the compound is found outside the AD; a compound outside the AD is an outlier. An outlier is an object that cannot be well described by the model; this situation arises in many occasions: the model could have some shortcomings and not be applicable to some instances, the outlier could be a very rare instance, an experimental error, etc.

If the model were a person deciding whether or not you could eat this evening, you could ask "him" the question: "knowing what you know and having seen how a few other people are fed, can you be trusted with such a big decision as whether I am to be given a meal tonight or not?" Here, the "few other people" are the training set and whether or not you are hungry and should be given a meal is the prediction. The purpose of the applicability domain is to determine whether the model is able to make a reliable prediction for you.

In which cases is the model not to be trusted? For the model in charge with tonight's meal, two problems might arise.

1. The structural outlier. Let us say that this model-person bases his decision exclusively on your corporal features (big, small, large, etc.). Then, if you are a big and large man, he will decide you must be very hungry. However, if you are the biggest and thinnest man he has ever seen, he will not know what to decide, since by his own experience big people need more food but slim persons do not eat very much. In this case, you could be discarded from the applicability domain as a structural outlier.

For molecules, structural outliers are descriptor representations that have values very different from the training set. An example of an applicability domain discarding structural outliers is the bounding box, where the molecules whose descriptors do not fall into a specified range (defined using the training set) are not included in the AD.

2. The activity outlier. Let us now say that you are very thin and small; the model will then decide you do not need an evening meal. The model has seen a lot of people just like you, and they were most of the time not hungry at all. However, you could be very hungry because you are a very sportive person; therefore, the model would be wrong not to feed you. This time again, you could be discarded from the applicability domain, but as an *activity* outlier; the model did not include some necessary features (such as sports) to correctly predict your *activity*, which is, in this case, your appetite.

For molecules, activity outliers are found by looking at activity variations of training set compounds in a specific region of the chemical space or by estimating the difference in predicted and experimental activity; the higher the variation, the higher the chances that the activity of a molecule found in this region will be incorrectly predicted. Therefore, we may consider that these regions are outside the applicability domain of the model. An illustration of these two types of outliers (structural and activity outliers) is given in Figure 9.1.

For GTM, we defined structure-dependent and activity-dependent applicability domains. Some of our methods directly discard molecules from the applicability domain that fail to fulfill a specific condition. On the other hand, other methods discard entire regions of the 2D map from the applicability domain instead of individual compounds, by finding which nodes on the map do not satisfy a specified requirement; the pre-

diction of molecules near those nodes will not be trusted. There are several ways to find molecules in discarded regions: simply find the nearest node for each molecule and establish whether it was discarded or not, or draw a delimited area around each node (Voronoi region, confidence ellipse, etc.) and capture molecules within the area. We mainly used the distance-based approach (with Euclidean distances).

9.2 Structure-Dependent AD

In this section, we define two types of structure-dependent AD using GTM, based on either the log likelihood or data density. These applicability domains can be used without activity information, *e.g.*, for simple visualization or clustering; but they can also be used for classification and regression models. The log likelihood-based AD discards individual test compounds from the applicability domain; the density-based AD discards nodes (regions on the 2D map) not fulfilling data density requirements.

9.2.1 Likelihood-Based

For GTM, the log likelihood of each molecule can be computed and used to determine whether a new molecule lies within the applicability domain; a related approach as been used by Bullen et al. for scatterometer data to discard outliers [94]. We define a value LF (likelihood factor), which is a percentile of the log likelihood distribution of training set molecules; for example, LF = 10% means that 90% of training compounds have a higher log likelihood than the value $\mathcal{L}_{LF}$. A query compound $\mathbf{t}_q$ with a log likelihood $\mathcal{L}_q$ that does not satisfy the following condition will be discarded from the applicability domain:

$$\mathcal{L}_q > \mathcal{L}_{LF} \tag{9.1}$$

For an application of this method, c.f. Chapter 16.

9.2.2 Density-Based

Another activity-independent AD easily defined with GTM is the density-based AD. First, a vector of cumulated responsibilities $\bar{r}$ is computed, where $\bar{r}_k$ is the value assigned to each node $\mathbf{x}_k$:

$$\bar{r}_k = \frac{\sum_n R_{kn}}{N} \tag{9.2}$$

then, a threshold value is fixed, and the node $\mathbf{x}_k$ (= region on the map) is discarded from the applicability domain if it does not obey the following rule:

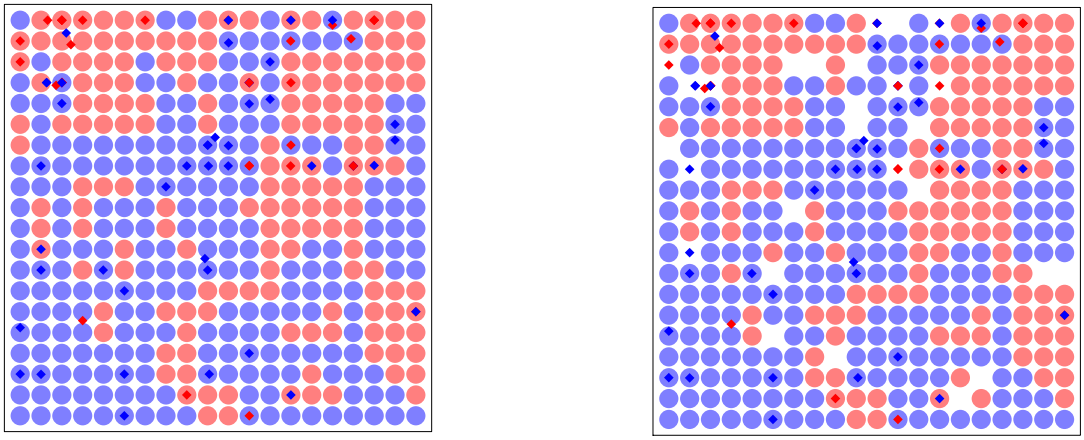
$$\bar{r}_k > threshold \tag{9.3}$$

where the threshold runs from 0 (all molecules within the AD) to 1 (all molecules outside the AD). This method is easily applicable to classification or regression models, and can be visualized on a 2D map easily in both cases: for classification models, by removing nodes outside the AD or using a transparency-based representation (Figure 7.1); for regression models, by shading regions of the activity landscape outside the AD (Figure 9.3).

9.3 Activity-Dependent AD

In this chapter, we introduce several new activity-based AD definitions for classification and regression using the GTM cartography.

9.4 Classification



(a) CPF=1, BAC=0.66, Coverage=100%

(b) CPF=3, BAC=0.76, Coverage=79%

Figure 9.2: BCRP (breast cancer resistance protein) inhibition classification map, where inhibitors are more likely to be found in red regions and non-inhibitors in blue regions; nodes are represented by large circles and tested compounds by points. The CPF applicability domain was applied on the right-hand map to remove nodes outside the applicability domain. The dataset was extracted from reference [95] and cleaned by Timur Gimadiev.

We introduce here 3 applicability domain definitions for classification models. The class-dependent density and class prevalence factor methods discard nodes from the applicability domain; in other words, some regions of the map will be considered as "no man's land". On the other hand, the class likelihood factor (CLF) defines which test compounds the model is able to predict and discards the prediction of a query compound that does not fulfill the CLF condition.

1. Class-dependent density. As explained in Chapter 7, a classification map is obtained by coloring each node of the map by the winning class c_{best}, which has the

highest conditional node probability $P(\mathbf{x}_k|c_{\text{best}})$ equal to the cumulated responsibilities within class c_{best}. GTM nodes (= regions on the map) within the AD fulfill the following requirement:

$$P(\mathbf{x}_k|c_{\text{best}}) > threshold \tag{9.4}$$

where the threshold runs from 0 (all molecules within the AD) to 1 (all molecules outside the AD). Therefore, this AD definition is very similar to the density-based AD, except we only take into account the density of training compounds belonging to the winning class. A representation of this applicability domain is given in Chapter 7, Figure 7.1. The problem with this approach is that it does not take into account information from other classes, and does not show the model's difficulty in choosing between one class or another, which is why we introduced the CPF (class prevalence factor) and CLF (class likelihood factor) approaches.

2. Class prevalence factor (CPF). The class prevalence factor is a simple value running from 1 (all compounds included in the applicability domain) to infinity, which must be inferior to the ratio of the node probability given the best class to the node probability given any over class:

$$CPF < \frac{P(\mathbf{x}_k|c_{\text{best}})}{P(\mathbf{x}_k|c_i)}, \forall c_i \neq c_{\text{best}} \tag{9.5}$$

In other words, the winning class c_{best} should prevail more than CPF times over all other classes in the node $\mathbf{x}_k$. If this is not the case, the node (region on the map) $\mathbf{x}_k$ is discarded from the applicability domain. We applied this AD definition and compared it to the bounding box method in Chapter 15; apparently, these two methods yield a similar performance. We also give an illustration of a CPF applicability domain in Figure 9.2 where discarded nodes are simply removed from the classification map; we applied the applicability domain on a model of BCRP (human breast cancer resistance protein) inhibition. CPF usually removes regions at the frontier between two classes, which is the main place where models "hesitate".

3. Class likelihood factor (CLF). The class likelihood factor method is based on the class entropy, which is a value defined for each compound measuring the disorder of class probabilities $P(c_i|\mathbf{t}_q)$ attributed to each query compound $\mathbf{t}_q$: if the probability of the winning class is far greater than any other, the entropy will be low, since the probability is concentrated in one place; however, if the probabilities are almost the same for all classes, the disorder will be great and the entropy high. The entropy E_q^{class} for a query compound $\mathbf{t}_q$ is defined as follows:

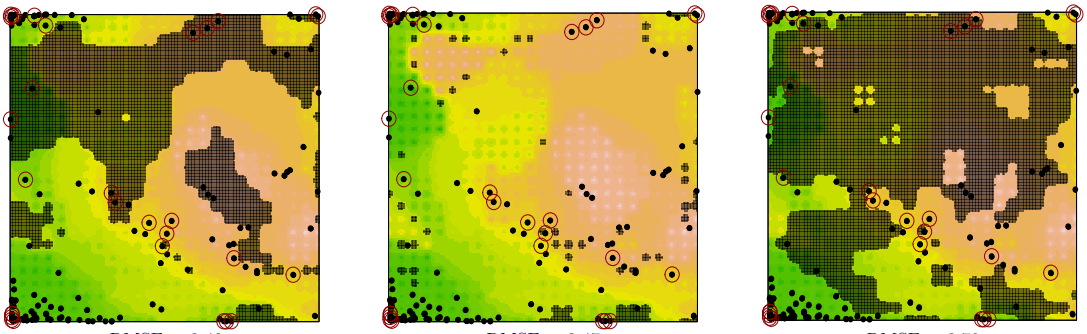
$$E_q^{\text{class}} = -\sum_i P(c_i|\mathbf{t}_q) \log(P(c_i|\mathbf{t}_q)) \tag{9.6}$$

Then, the condition for lying within the applicability domain is the following for the query compound $\mathbf{t}_q$:

$$CLF > \frac{E_q^{\text{class}}}{E_{max}^{\text{class}}} \tag{9.7}$$

where E_{max}^{class} is the highest class entropy, which occurs when every probability $P(c_i|\mathbf{t}_q)$ is equal to $\frac{1}{N_c}$, where N_c is the number of classes; i.e, when all conditional class probabilities are equal and no class wins. It should be noted that, in this case, we do not discard nodes from the applicability domain but establish that some queries cannot be correctly predicted. Therefore, this AD definition cannot be directly visualized as the CPF method. However, we also applied this method to classification models in Chapter 15 and its efficiency was close to that of CPF.

9.5 Regression



(a) Landscape Difference. **(b)** Activity Variance. **(c)** Density.

Figure 9.3: Three different applicability domains applied to a model predicting the binding constant of Gd^{3+}-ligand complexes; the landscape represents the binding constant values on the map, and the shaded areas are outside the applicability domain; circles are drawn around compounds with pK RMSE > 4.7 (20% worst predictions).

We defined two activity-based applicability domains, which discard nodes with high training set activity variation (activity variance method) or nodes with a high difference between experimental and predicted landscapes (landscape difference method).

1. Activity variance method. The activity variance σ_k^2 is estimated in each node $\mathbf{x}_k$ using the following formula:

$$\sigma_k^2 = \frac{\sum_{n=1}^{N} R_{kn}(a_n - \bar{a}_k)^2}{\sum_{n=1}^{N} R_{kn}} \tag{9.8}$$

where σ_k^2 is the activity variance in node $\mathbf{x}_k$, a_n the experimental activity of compound

$\mathbf{t}_n$, $\bar{a}_k$ the average activity in node $\mathbf{x}_k$, and R_{kn} the responsibility of node $\mathbf{x}_k$ for molecule $\mathbf{t}_n$.

2. Landscape difference. The landscape difference method involves computing the following differences between the predicted landscape values $\bar{a}(\text{pred})_k$ at each node $\mathbf{x}_k$ and the experimental landscape values $\bar{a}(\text{exp})_k$:

$$diff_k = abs(\bar{a}(\text{pred})_k - \bar{a}(\text{exp})_k) \tag{9.9}$$

$\bar{a}(\text{exp})_k$ is computed using the experimental activities of the training set and $\bar{a}(\text{pred})_k$ using the predicted activities:

$$\bar{a}(\text{exp})_k = \frac{\sum_{n=1}^{N} R_{kn} \times a(\text{exp})_n}{\sum_{n=1}^{N} R_{kn}} \tag{9.10a}$$

$$a(\text{pred})_n = \sum_{k=1}^{K} R_{kn} \times \bar{a}(\text{exp})_k \tag{9.10b}$$

$$\bar{a}(\text{pred})_k = \frac{\sum_{n=1}^{N} R_{kn} \times a(\text{pred})_n}{\sum_{n=1}^{N} R_{kn}} \tag{9.10c}$$

A node $\mathbf{x}_k$ will be within the applicability domain if:

$$diff_k < threshold \tag{9.11}$$

We experimented three AD for regression models (activity variance, landscape difference, and density) on a dataset of complexants of Gd^{3+}, where the modelled activity is the ligand-metal complex stability constant, in Figure 9.3. The activity landscape was built using training set molecules; the points on the maps are test set molecules, and the most incorrectly predicted molecules (20%) are highlighted; the goal of the operation was to diminish the prediction error (RMSE) without discarding too many test molecules from the applicability domain (coverage). We did not systematically evaluate the performance of these AD definitions in any other practical application; the subject of supervised AD for GTM regression models could be further investigated. However, the landscape difference method shows promising results, whereas the activity variance method mainly discards the most populated regions, resulting in a low coverage.

Chapter 10

Chemical Libraries Analysis

In this chapter, we show how to compare and analyze large chemical libraries using GTM. A part of this analysis consists in building activity and descriptor landscapes, as described in Chapter 5. However, the key concept of our "big data" analysis is to use probability distributions for whole data subsets (libraries) instead of considering each element (molecule) one by one. We use these distributions to estimate the similarity of libraries and their dispersion in descriptor space. An application of these methods to a set of 37 chemical libraries (more than 2 million compounds) can be found in Chapter 17.

10.1 Cumulated Responsibilities

Our analysis of large libraries is based on cumulated responsibilities. Since responsibilities are assigned to every compound mapped on the GTM map, we can compute cumulated responsibilities $\bar{\mathbf{R}}$ for each node $\mathbf{x}_k$ and library c_i:

$$\bar{R}_{ik} = \frac{\sum_n R_{kn}(c_i)}{N_{c_i}} \tag{10.1}$$

where N_{c_i} is the population within the library. Therefore, instead of considering each library c_i as a set of compounds, we represent it by a single K-dimensional vector of cumulated responsibilities $\bar{\mathbf{r}}_i$, where K is the number of nodes on the map.

10.2 Library Diversity

A library's diversity can be assessed by computing Shannon's entropy [96]:

$$E(c_i) = -\sum_k \bar{R}_{ik} \log(\bar{R}_{ik}) \tag{10.2a}$$

$$E_{Norm}(c_i) = \frac{E(c_i)}{\log(K)} \tag{10.2b}$$

where $E_{\text{norm}}(c_i)$ is the normalized entropy of library c_i. This measure is an indicator of the dispersion of a library on the 2D map.

10.3 Comparing Landscapes

Landscapes $\{\bar{\mathbf{a}}(c_i)\}$ can be computed for each library c_i. These landscapes can be employed to compare libraries according to a specific property. We introduce a measure coined RLE or relative landscape elevation, to measure the percentage of the map where the landscape of a library has higher values than a landscape of reference:

$$RLE_i = \frac{1}{K} \sum_{k=1}^{K} H(\bar{a}_k(c_i) - \bar{a}_k(\text{reference})) \times 100\% \tag{10.3}$$

where H is the Heaviside step function ($H(x) = 0$ if $x < 0$ and $H(x) = 1$ if $x > 0$), RLE_i is the relative landscape elevation (for a specific property) for library c_i, $\bar{a}_k(c_i)$ the landscape value for library c_i at node $\mathbf{x}_k$ and $\bar{a}_k(\text{reference})$ the reference landscape value. Typically, the reference could be the landscape computed using all libraries. Relative landscape elevations can be obtained for each library and plotted in a bar graph (see Chapter 17). A library has higher property values than the reference database on $RLE\%$ of the map; for example, if $RLE = 100\%$, the library has higher property values than the reference everywhere on the map.

10.4 Library Similarity

Libraries can be compared using their cumulated responsibilities vectors $\bar{\mathbf{r}}_i$. Some possible similarity or distance measures $S(c_i, c_j)$ include:

1. The Battacharyya similarity:

$$S_{Bhattacharyya}(c_i, c_j) = \sum_k \sqrt{\bar{R}_{ik} \times \bar{R}_{jk}} \tag{10.4}$$

2- The Euclidean distance:

$$S_{Euclidean}(c_i, c_j) = \sqrt{\sum_k \left|\bar{R}_{ik} - \bar{R}_{jk}\right|^2} \tag{10.5}$$

3. The Tanimoto similarity:

$$S_{Tanimoto}(c_i, c_j) = \frac{\sum_k \bar{R}_{ik}\bar{R}_{jk}}{\sum_k \bar{R}_{ik}^2 + \sum_k \bar{R}_{jk}^2 - \sum_k \bar{R}_{ik}\bar{R}_{jk}} \tag{10.6}$$

These similarities and distances can also be used in the initial descriptor space to compare molecules. If the descriptors are binary, the Tanimoto similarity between two compounds I and J is usually used in chemoinformatics:

$$T(I, J) = \frac{N(1,1)}{N(0,1) + N(1,0) + N(1,1)} = \frac{N(1,1)}{D - N(0,0)} \tag{10.7}$$

and can be used to compute an average distance between two libraries:

$$T(c_i, c_j) = \frac{1}{N_{c_i} N_{c_j}} \sum_{I \in c_i} \sum_{J \in c_j} T(I, J) \tag{10.8}$$

However, evaluating distances in the initial space is much more costly than using cumulated responsibilities, since in the initial space all distances between all molecules must be computed. On the other hand, by using cumulated responsibilities, comparing two libraries is limited to computing the similarity between two vectors of cumulated responsibilities.

10.5 Meta-GTM

We introduced the μGTM or meta-GTM concept while investigating chemical libraries (thanks to Dr. D. Horvath for finding a better name than "GTM^2"). The μGTM map is a GTM map with a second level of abstraction: after training a first map with data instances, the cumulated responsibilities vectors are computed for each library (or data subset), so that each library is represented by a vector of cumulated responsibilities; then, these vectors are used as if they were data instances to perform a second GTM and finally produce the μGTM map. Points on the 2D μGTM map do not represent molecules, but whole libraries. This method is useful when multiple data subsets or libraries must be compared. The cumulated responsibilities can be seen as library descriptors; other types of library descriptors can also be used, such as landscape vectors or RLE values. To see applications of these approaches, see Chapter 17.

Chapter 11

Stargate GTM (S-GTM)

11.1 Introduction

In this chapter, we introduce the Stargate version of the generative topographic mapping (S-GTM) and its use as a supervised regression method. In conventional 1-space GTM, the D-dimensional descriptor space is reduced to a 2-dimensional space; with Stargate GTM, two separate spaces (A-dimensional and B-dimensional), *e.g.*, the space of chemical descriptors and the space of experimental properties, are optimized in parallel. With a Stargate GTM model, it is possible to map data described only in one of the spaces (*e.g.*, the space of chemical descriptors) into the other space (*e.g.* the space of properties). The original idea of S-GTM (using joint responsibilities to train a model) originated from Prof. I. I. Baskin; the algorithm was developped and implemented during this thesis. The method was first called "combined GTM" before the name "Stargate GTM" was adopted thanks to Prof. Varnek; however, the term "combined GTM" is still used in the command-line tool. In this chapter, we first describe the new Stargate GTM algorithm and the methodology to train and test a model. Then, we build Stargate models to predict coordinates in an 8-dimensional affinity space or in individual 1-dimensional spaces corresponding to pKi values for 8 different targets, based on a dataset of 1325 molecules from ChEMBL [97]. In the third section, we compare the performance of Stargate GTM model to conventional GTM models using four datasets: Lu binders (Lu dataset), thrombin inhibitors (Threx), a solubility dataset (LogS), and compounds from the ChEMBL database (ChEMBL). Then, we investigate how the performance of Stargate GTM models could be affected by weights applied to each space, and finally study the effect of the correlation of properties to be predicted.

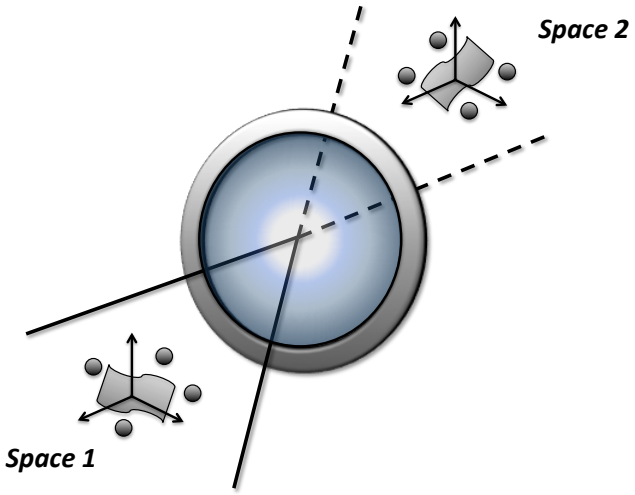


Figure 11.1: Molecules can jump through the GTM "Stargate" from one vector space, where they are described by a specific set of variables, to another vector space, with a different set of variables. For example, Space 1 could be the space of structural descriptors, and Space 2 the space of experimental properties. A molecule, by passing through the gate, could jump from Space 1 to Space 2 and vice versa. The operation is not symmetric: two different manifolds are used to travel to the 2 different spaces.

11.2 Stargate GTM

11.2.1 Algorithm

The GTM can reduce a D-dimensional data space to a 2D latent space. Stargate GTM uses a combined model trained on two spaces with dimensionalities A and B as a "gate" from one space to another (cf. Figure 11.1).

One molecule can therefore jump through the gate form one space (e.g. the space of structural descriptors) to another (e.g. the space of experimental properties). For this purpose, S-GTM optimizes two manifolds for the two different spaces separately, and combines the probability density functions (PDFs) to obtain a combined probability distribution for both spaces. Thus, two different manifolds are built, which may be used to project new molecules from one space to another. The Stargate GTM workflow is divided into two steps:

Training stage:

Train a model using Space 1 and Space 2 descriptors: we obtain 2 separate manifolds for Space 1 and Space 2.

Test stage:

1. Projection of A-dimensional Space 1 data to the Stargate GTM 2D space.

2. Mapping of the 2D coordinates from GTM 2D space to the B-dimensional Space 2.

11.2.1.1 Training Stage

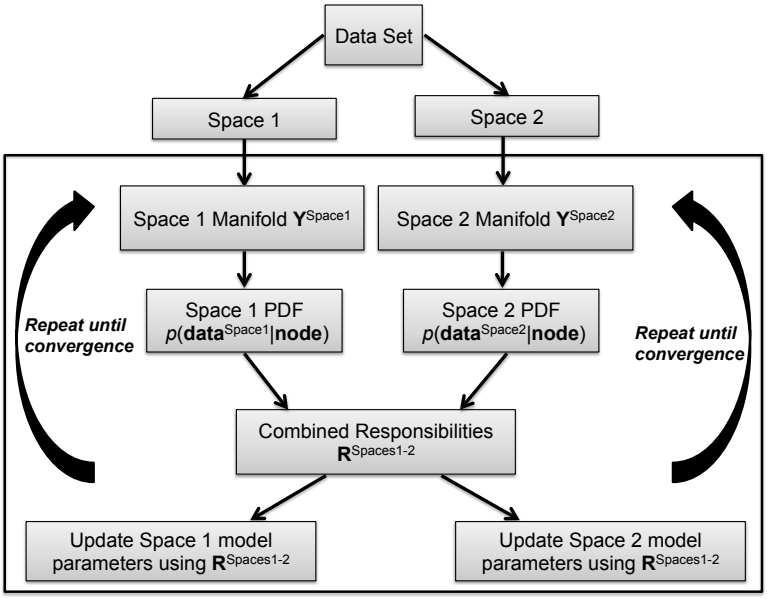


Figure 11.2: S-GTM training stage workflow.

At the training stage (Figure 11.2), two different probability density functions (PDFs) $p(\mathbf{t}^{\text{Space1}}|\mathbf{x}_k, \mathbf{W}^{\text{Space1}}, \beta^{\text{Space1}})$ and $p(\mathbf{t}^{\text{Space2}}|\mathbf{x}_k, \mathbf{W}^{\text{Space2}}, \beta^{\text{Space2}})$ are considered for Space 1 and Space 2, respectively. They can be computed using two different manifolds $\mathbf{Y}^{\text{Space1}}$ and $\mathbf{Y}^{\text{Space2}}$ as well as two different inverse variances β^{Space1} and β^{Space2}. Posterior probabilities (responsibilities) for Space 1 and Space 2 $\mathbf{R}^{\text{Space1}}$ and $\mathbf{R}^{\text{Space2}}$ are computed during the expectation step:

$$R_{kn}^{\text{Space1}} = \frac{p(\mathbf{t}_n^{\text{Space1}}|\mathbf{x}_k, \mathbf{W}^{\text{Space1}}, \beta^{\text{Space1}})}{\sum_{k'} p(\mathbf{t}_n^{\text{Space1}}|\mathbf{x}_{k'}, \mathbf{W}^{\text{Space1}}, \beta^{\text{Space1}})} \tag{11.1a}$$

$$R_{kn}^{\text{Space2}} = \frac{p(\mathbf{t}_n^{\text{Space2}}|\mathbf{x}_k, \mathbf{W}^{\text{Space2}}, \beta^{\text{Space2}})}{\sum_{k'} p(\mathbf{t}_n^{\text{Space2}}|\mathbf{x}_{k'}, \mathbf{W}^{\text{Space2}}, \beta^{\text{Space2}})} \tag{11.1b}$$

as well as combined responsibilities:

$$R_{kn} = \frac{p(\mathbf{t}_n^{\text{Space1}}|\mathbf{x}_k, \mathbf{W}^{\text{Space1}}, \beta^{\text{Space1}}) \times p(\mathbf{t}_n^{\text{Space2}}|\mathbf{x}_k, \mathbf{W}^{\text{Space2}}, \beta^{\text{Space2}})}{\sum_{k'} p(\mathbf{t}_n^{\text{Space1}}|\mathbf{x}_{k'}, \mathbf{W}^{\text{Space1}}, \beta^{\text{Space1}})^{w^{\text{Space1}}} \times p(\mathbf{t}_n^{\text{Space2}}|\mathbf{x}_{k'}, \mathbf{W}^{\text{Space2}}, \beta^{\text{Space2}})^{w^{\text{Space2}}}} \tag{11.2}$$

where w^{Space1} and w^{Space2} are weight factors attributed to each space, ranging from 0 to 1: $[0 \leq w^{\text{Space1}} \leq 1, w^{\text{Space2}} = 1 - w^{\text{Space1}}]$. These combined responsibilities $\mathbf{R}$

will then be used to adjust the shape of the manifolds $\mathbf{Y}^{\text{Space1}}$ and $\mathbf{Y}^{\text{Space2}}$ during the maximization step. This procedure is repeated until convergence of the log likelihood function.

11.2.1.2 Test Stage

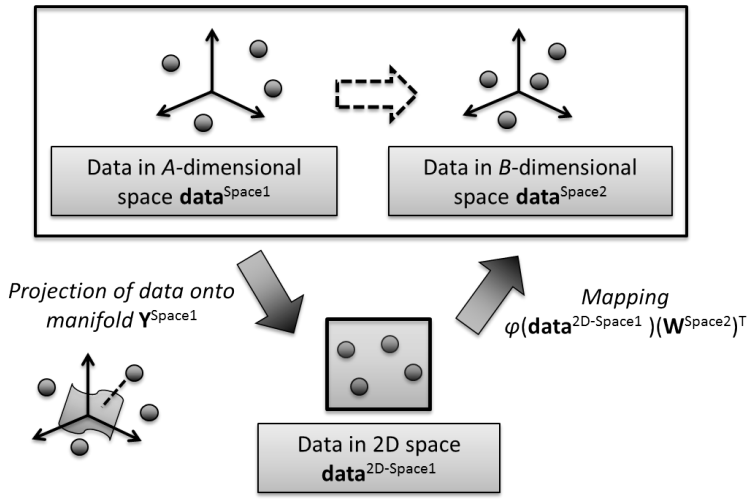


Figure 11.3: S-GTM test stage workflow.

S-GTM test stage. As shown in Figure 11.3, at the test stage, data in Space 1 (descriptor space) $\mathbf{T}^{\text{Space1}}$ is mapped into Space 2 (activity space) to obtain $\hat{\mathbf{T}}^{\text{Space2}}$. The coordinates of a data point in Space 2 correspond to its properties (or activities). The mapping from Space 1 to Space 2 $\mathbf{T}^{\text{Space1}} \rightarrow \hat{\mathbf{T}}^{\text{Space2}}$ proceeds in two steps. First, A-dimensional data points from Space 1 are projected into the 2D latent space:

$$\mathbf{T}^{\text{Space1}} \rightarrow \mathbf{X}(\mathbf{T}^{\text{Space1}}) \tag{11.3}$$

then, they are mapped from the latent space into Space 2:

$$\mathbf{X}(\mathbf{T}^{\text{Space1}}) \rightarrow \hat{\mathbf{T}}^{\text{Space2}} \tag{11.4}$$

The projection $\mathbf{T}^{\text{Space1}} \rightarrow \mathbf{X}(\mathbf{T}^{\text{Space1}})$ is performed by computing the distances in Space 1 between the data points $\mathbf{T}^{\text{Space1}}$ and manifold points $\mathbf{Y}^{\text{Space1}}$. Then, these distances are used to compute the associated PDFs and responsibilities, from which the 2D coordinates of the nth molecule $\mathbf{x}(\mathbf{t}_n^{\text{Space1}})$ can be obtained:

$$\mathbf{x}(\mathbf{t}_n^{\text{Space1}}) = \sum_k^K \mathbf{x}_k R_{kn}^{\text{Space1}} \tag{11.5}$$

The first step of the "reverse" mapping of the nth molecule from the latent space to the B-dimensional Space 2 consists in applying radial basis functions to $\mathbf{x}(\mathbf{t}_n^{Space1})$:

$$\phi_m(\mathbf{x}(\mathbf{t}_n^{Space1})) = \exp\left(-\frac{\left\|\mathbf{x}(\mathbf{t}_n^{\text{Space1}}) - \boldsymbol{\mu}_m\right\|^2}{2\sigma^2}\right) \tag{11.6}$$

where $\boldsymbol{\mu}_m$ is the center of the mth RBF and σ a parameter tunable by the user; these functions are used together with a parameter matrix $\mathbf{W}^{\text{Space2}}$ to map the 2D data points into Space 2:

$$\hat{t}_{nd}^{\text{Space2}} = \sum_{m}^{M} \phi_m(\mathbf{x}(\mathbf{t}_n^{\text{Space1}}))W_{dm}^{Space2} \tag{11.7}$$

11.2.2 Prediction with S-GTM

Prediction with S-GTM can be done in the same way as with conventional GTM, by using an activity landscape or a k-NN approach. However, if Space 1 is the descriptor space and Space 2 the activity space, coordinates of molecules in an activity space, *i.e.*, predictions, can be directly obtained by mapping from descriptor space to activity space, without the need of an activity landscape; this is the prediction method we used in this chapter. However, this method is valid only if responsibilities $\mathbf{R}^{\text{Space1}}$ have one mode per molecular structure, so that the posterior mean will be a good summary of the distribution; in our case, it was indeed usually the case when mapping from descriptor space to activity space. On the other hand, when the process is performed the other way around (from activity space to descriptor space), the responsibilities $\mathbf{R}^{\text{Space2}}$ usually have several modes for only one activity corresponding to several possible structures; in this case, computing the mean is not sufficient. However, several ways may be used to find the corresponding structures, which will not all be explored here; we will only illustrate one of them, which consists in selecting the nodes $\{\mathbf{x}_k\}$ with the highest responsibility values $\left\{R_{kn}^{\text{Space2}}\right\}$ for the nth activity, and simply retrieve the training set compounds on the 2D map that can be found within delimited areas drawn around each of these nodes.

11.3 Application 1: Prediction of 8 Affinities

11.3.1 Data and Descriptors

For S-GTM, both structural (molecular descriptors) and activity data for each molecule are required. However, it is sometimes rather difficult to find enough experimental data to fill completely the activity matrix. In this work, we used a dataset containing 1325 ligands of 8 different rhodopsin-like GPCR receptors extracted from the ChEMBL

database. The set was chosen to provide experimental pKi values for all molecules for one of these receptors (Dopamine D2). For the others, missing experimental affinities were completed by values theoretically predicted by SVM models reported in [98]. The dataset was randomly split into a training set of 883 compounds and a test set of 442 compounds. The modeling was performed using ISIDA atom-centered fragments IIA(1-5)_P [20, 99] defining the descriptor space. These descriptors provide with the best SVM models for dopamine receptor D2 affinity, for which experimental values were available for all 1325 compounds. The descriptors were generated by ISIDA Fragmentor v. 2013 program [99]. Notice that descriptors with more than 90% null values were discarded. The target names and number of experimental and predicted affinities available are given in Table 11.1.

Table 11.1: The Affinity dataset contains 1325 molecules with affinities for 8 targets. The target names, ChEMBL IDs and short IDs used in this study are given in the table. Only some compounds had experimentally measured pKi values of affinity for a given target. The number of experimental (exp) and predicted (pred) affinities is indicated.

ChEMBL ID	Short ID	Target Name	Exp.	Pred.
1867	A2a	Alpha-2a adrenergic receptor	22	1303
271	D2	Dopamine D2 receptor	1325	0
234	D3	Dopamine D3 receptor	684	641
219	D4	Dopamine D4 receptor	335	970
214	S1a	Serotonin 1a (5-H1a) receptor	229	1096
224	S2a	Serotonin 2a (5-H2a) receptor	187	1138
3155	S7	Serotonin 7 (5-HT7) receptor	20	1305
228	ST	Serotonin transporter	58	1267

11.3.2 Methods

In this section, an "Affinity" dataset was used to predict 8 affinities towards targets listed in Table 2 11.1. We compared the performance of Stargate GTM to the implementation of Random Forests in WEKA [100] (with a small default number of tree = 10) and conventional GTM.

For conventional GTM, the internal/external validation was performed as explained in Chapter 6; the best parameters were selected in a cross-validation procedure, by keeping the number of RBF centers $M = 400$, the number of grid nodes $K = 400$, and choosing the regularization coefficient λ in [0.01, 0.1, 1, 10, 100] and the width factor w in [0.25, 0.5, 1, 1.5, 2]. The best parameters were then used to build a model, which was used to predict the 8 affinities for compounds in the external test set. For conventional GTM, we used the GTM activity landscape method for predictions, which

yielded a better performance than the GTM k-NN approach in this case.

For S-GTM, the parameter selection was the same; however, it was not necessary to make an optimization for each property. Only one optimization is required for S-GTM, since all properties are predicted at the same time. Therefore, 25 Stargate GTM models were built by varying the w and λ parameters, the weight parameter was fixed at 50% for both spaces, and the 3-fold cross-validated Q^2 was also obtained each property. However, in this case, we had only one set of parameters to find for all properties, in opposition to conventional GTM, where we optimized one model per property. Thus, the Q^2 values for all properties had to be taken into account: the model with the largest proportion of properties with a Q^2 higher than 0.50 was selected. If two models had the same proportion of $Q^2 > 0.50$, then the models with the highest Q^2 averaged across all properties won. Then, the external test set was sent through this optimized gate from the descriptor space into the property space, and we obtained directly predictions for all properties and the associated determination coefficients.

S-GTM/8D and S-GTM/1D protocols. We employed two different Stargate GTM approaches to predict the 8 affinities: with S-GTM/8D, we used the whole 8-dimensional affinity space to build a model and be able to predict all 8 affinities at the same time. With S-GTM/1D, for each affinity separately, we used a 1-dimensional affinity space (corresponding to the affinity for one single target) to build a model and predict only one activity at a time. Thus, for both S-GTM/8D and Stargate S-GTM/1D, we built a supervised model, which was trained with the knowledge of activity values of a training set. With S-GTM/1D, we simply predict one activity; with S-GTM/8D, we predict the whole 8-dimensional activity profile. Only the parameters of Stargate GTM 8D were optimized and used to build both S-GTM/8D and S-GTM/1D models.

11.3.3 Results and Discussion

As mentioned in the foregoing section, we used an 8-dimensional (entire profile, with prediction-filled missing values) and a 1-dimensional affinity space (8 models for 8 separate affinities) for the S-GTM modeling of the Affinity dataset.

The S-GTM model allows us to project data into the 2D latent space either from descriptor space (with responsibilities $\mathbf{R}^{\text{Space1}}$) or from activity space (with responsibilities $\mathbf{R}^{\text{Space2}}$). These two data projections look very different for S-GTM/8D and S-GTM/1D protocols in Figure 11.4. Indeed, in S-GTM/8D calculations, data projection from the descriptor space into the 2D latent space results in a set of points scattered on the map. On the other hand, data projection from 1D activity space into the 2D space with S-GTM/1D results in a sort of one-dimensional curve embedded in the latent space. This could be explained by significant differences between $\mathbf{R}^{\text{Space1}}$ and $\mathbf{R}^{\text{Space2}}$ distributions, which define positions of data points (mean of the distributions).

In fact, the $\mathbf{r}_n^{\text{Space1}}$ distribution for one molecule is in most cases unimodal (Figures 6c), whereas the $\mathbf{r}_n^{\text{Space2}}$ distribution is usually multimodal. The former reflects the fact that a molecule is characterized by specific values of descriptors and activities, whereas the latter shows that structurally different molecules might have similar activity values. In particular, "inactives" are not subject to any structural constraints - they might be scattered anywhere in the descriptor space.

This is a consequence of the intrinsic asymmetry of QSAR modeling and chemical similarity concept: similar molecules should have similar activities, but similar activities could correspond to quite different chemical structures. Thus, passing a molecule through the "gate" (S-GTM model) from the descriptor to the activity space, S-GTM predicts its activities. On the other hand, passing an activity query (a set of specified activity values) from the activity to the descriptor space, S-GTM performs an inverse QSAR analysis: it finds areas in the descriptor space in which other molecules might have the specified activity values.

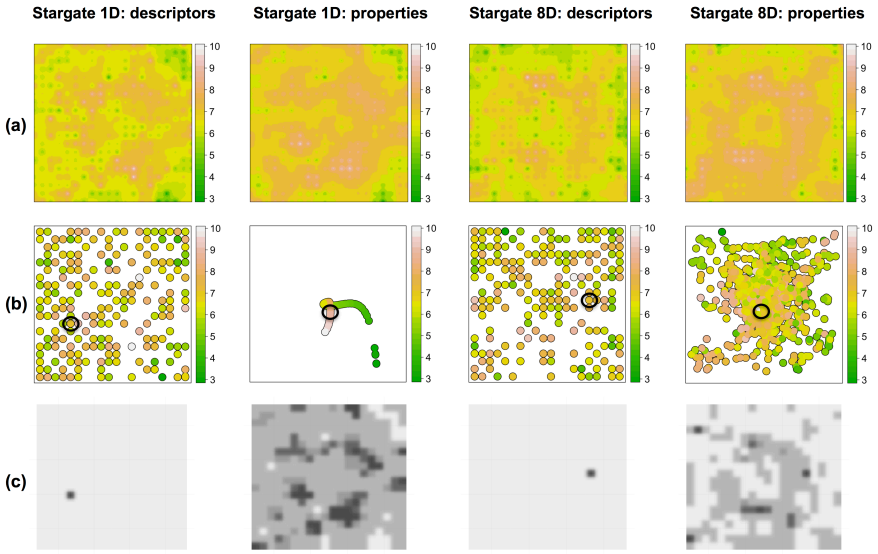


Figure 11.4: Maps obtained with the S-GTM models using responsibilities $\mathbf{R}^{\text{Space1}}$ (descriptors) and $\mathbf{R}^{\text{Space2}}$ (properties). $\mathbf{R}^{\text{Space1}}$ and $\mathbf{R}^{\text{Space2}}$ result from mapping the data from the descriptors and activity spaces, respectively, to the 2D latent space. Three different representations are given: (a) activity landscapes of D2 affinity, (b) average position of data points, and (c) responsibilities of a data point selected in (b).

We use the term "inverse QSAR" to denote only the first step (from activities to molecular descriptors) of what is actually known as "inverse QSAR" in literature. S-GTM does not directly generate new structures from scratch; it only finds areas where known structures can exhibit specific activities. Those areas correspond to different modes of the multimodal distribution in the descriptor space. They are depicted by the

most intense dark grey spots in Figure 11.4.

Several important observations can also be made from close inspection of Figure 11.4. First, the mode position of the unimodal distribution of the data from the descriptor space coincides exactly with one of the modes of the multimodal distribution conditioned on data from the activity space (second column). This means that a molecule, having traveled through the gate to the activity space, and returned from there through the same gate back to the descriptor space, is reconstructed. The second important observation is that the areas corresponding to different modes of this distribution have the same color in the affinity landscapes as the traveling molecule. This means that after returning from the activity space, the molecule is accompanied by several other molecules with a similar activity value or activity profile. The third observation is that the number of those companion molecules is smaller for 8D affinity space than for 1D affinity space. This could be explained by a much lower probability to find a molecule with similar affinities for several targets than for a single target.

It is interesting to note that D2 activity landscapes obtained by projecting data (descriptor values) from the descriptor space to the 2D latent space and by projecting data (activity values) from the affinity space to the 2D latent space are very similar (see Figure 11.4). The Pearson correlation coefficient between descriptor and activity landscapes is 75% for both S-GTM/8D and S-GTM/1D protocols.

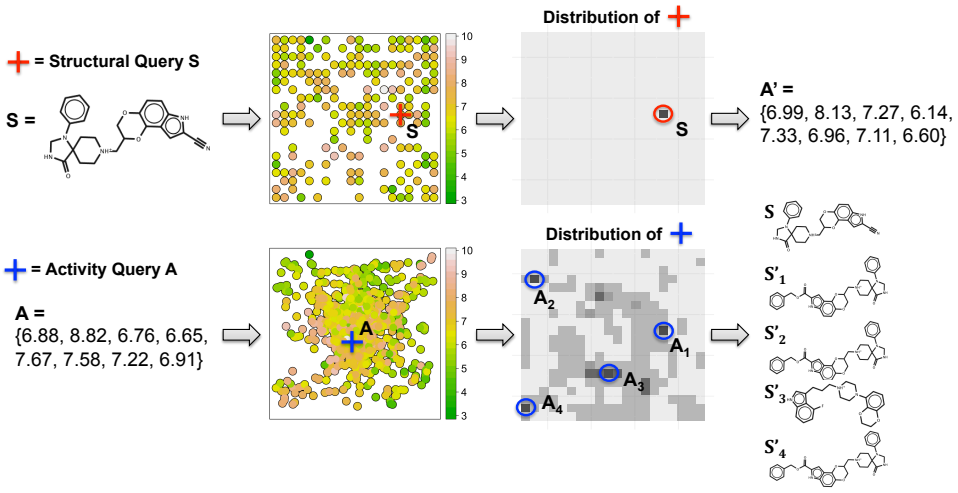
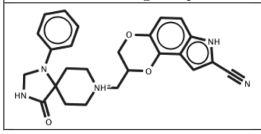


Figure 11.5: Activity prediction (conventional QSAR) and structures retrieval (inverse QSAR) with S-GTM. Using a structural query as input, S-GTM predicts an activity profile (top), whereas by using an activity query S-GTM retrieves several structures whose activities are similar to the query (bottom). **S** was retrieved in the same region ($\mathbf{A}_1$) as $\mathbf{S'}_1$. See Figure 11.6.

In Figure 11.5, we show how a single S-GTM model can be used in 2D for both conventional and inverse QSAR. For this, we used maps built with the S-GTM/8D protocol (cf. Figure 11.4 for the different representations). In a conventional QSAR

(a)

Structural query S	Activity (exp) A	Activity (pred) A'
	{6.88, 8.82, 6.76, 6.65, 7.67, 7.58, 7.22, 6.91}	{6.99, 8.13, 7.27, 6.14, 7.33, 6.96, 7.11, 6.6}

(b)

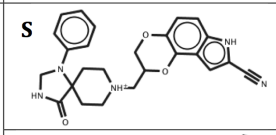
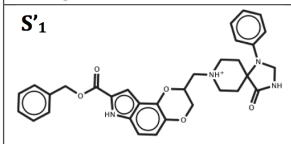
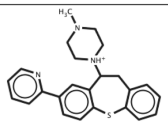
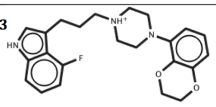
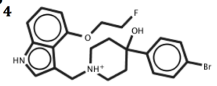
Activity query A	Retrieved structures	Activity (exp)
{6.88, 8.82, 6.76, 6.65, 7.67, 7.58, 7.22, 6.91}	**S** 	{6.88, 8.82, 6.76, 6.65, 7.67, 7.58, 7.22, 6.91}
	S'1 	{6.85, 8.23, 7.06, 6.66, 6.94, 7.56, 6.92, 7.14}
	S'2 	{6.87, 8.72, 7.27, 6.75, 7.45, 7.46, 7.3, 7.34}
	S'3 	{6.84, 8.09, 6.77, 7.08, 7.87, 7.74, 7.2, 6.89}
	S'4 	{6.78, 7.55, 6.07, 6.17, 7.04, 7.46, 6.66, 7.27}

Figure 11.6: Table coupled with Figure 11.6: (a) structural query **S** and corresponding experimental and predicted activity profiles **A** and **A′**; (b) activity profile query **A** and the structures **S′** found using S-GTM, with their associated experimental activity profiles.

procedure, an activity profile $\mathbf{A}'$ is predicted from a structure $\mathbf{S}$; as shown in Figure 11.5, a structure with a unimodal distribution will have only one predicted activity profile $\mathbf{A}'$. On the other hand, for the inverse QSAR procedure where the goal is to find structures $\mathbf{S}'$ corresponding to an activity profile $\mathbf{A}$, the probability distribution of the query $\mathbf{A}$ on the 2D map is multimodal, so that modes of the distribution must be selected to find structures $\{\mathbf{S}'_i\}$ in the corresponding regions. Therefore, from $\mathbf{S}$ to $\mathbf{A}'$ (structure to activity), there is most of the time a one-to-one correspondence, but from $\mathbf{A}$ to $\{\mathbf{S}'_i\}$ (activity to structure), multiple answers to the query are possible. The more dispersed the responsibilities, and the larger the number of modes of the activity profile distribution in 2D, the more diverse will be the structures found by S-GTM. In Figure 11.4, we may see that the distribution of a single activity covers a larger portion of the map than the 8D activity profile, which confirms the simple fact that

they are more compounds sharing the same single activity than compounds sharing a whole activity profile. In Table 4, we show the queries **S** for QSAR and **A** for inverse QSAR, as well as the answers given by S-GTM ($\mathbf{A}'$ for the predicted profile and $\{\mathbf{S}'_i\}$ for the predicted structures).

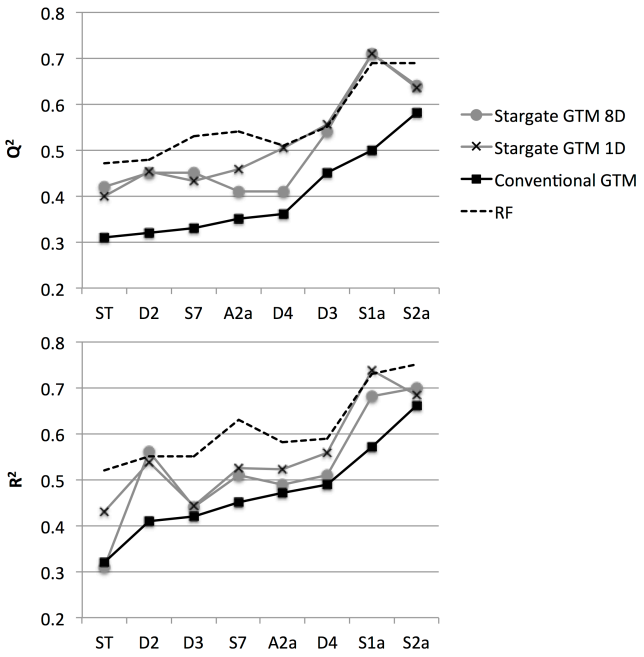


Figure 11.7: Predictive performance of Random Forests (RF), GTM and S-GTM models for 8 different activities in the Affinity dataset assessed in 3-fold cross-validation on the training set (top), and on the external test set (bottom). All GTM models were obtained with optimized sets of parameters. The data points were sorted in ascending order of conventional GTM determination coefficients Q^2 and R^2.

The model performances Q^2 (internal performance assessed in 3-CV on the training set) and R^2 (external performance assessed on the external test set) for conventional GTM, S-GTM/1D, S-GTM/8D and RF models are given in Figure 11.7, for each affinity listed in Figure 11.1.

The internal and external predictive performances are very close. The only affinities predicted with an internal performance $Q^2 > 0.5$ were S1a (affinity for Serotonin 1a receptor) and S2a (affinity for Serotonin 2a receptor), for which most models achieved Q^2 and R^2 values close to 0.70. We can see that S-GTM models always outperform conventional GTM models, although they are still less predictive than the Random Forests models. We believe that the S-GTM performance could be improved by optimizing weight factors w^{Space1} and w^{Space2} used to compute the combined responsibilities **R**.

11.3.4 Conclusion

With S-GTM, an activity profile can be predicted by mapping an object (chemical structure) from descriptor space to activity space; on the other hand, by mapping from the activity space to the descriptor space, S-GTM delineates areas in descriptor space populated by molecules possessing specific activity values, in this way performing an inverse QSAR analysis. As a supervised learning method, S-GTM regression should outperform conventional GTM models. This is clearly observed in the activity profile prediction challenge, although S-GTM predictions are slightly inferior to those obtained with the popular Random Forests method. We believe that the S-GTM performance could be improved by optimizing the weight factors characterizing relative contributions of each space at the training stage. Since S-GTM builds simultaneously two manifolds, two different data projections from the two initial data spaces (descriptors and activities) can be visualized. The probability distribution of a data point projected from the descriptor space into the latent space is in most cases unimodal, whereas the probability distribution of a data point projected from the activity space into the latent space is often multimodal. These multimodal and unimodal distributions from descriptor or activity space illustrate the fact that a chemical structure characterized by a set of molecular descriptors has a particular activity profile, whereas a given activity profile may correspond to several different chemical structures.

11.4 Application 2: Prediction of MOE 2D Descriptors

11.4.1 Data and Descriptors

In order to investigate the Stargate GTM approach, we must define two different spaces, one for properties, the other for descriptors. In this section, we used the ChEMBL [97], LogS, Lu and Thrombin datasets described in Table 11.2, and computed ISIDA fragments for the descriptor space and MOE 2D descriptors for the property space. We also

Table 11.2: Datasets used to build conventional and Stargate GTM models. The ChEMBL, LogS, Lu, and Thrombin datasets were used for methodological tests, to build models predicting MOE properties.

Dataset	Training set	Test set
ChEMBL	500	500
LogS	818	817
Lu	95	94
Thrombin	107	106

used the LogS [101, 102, 103], Lu [104], and Thrombin [105] datasets when studying

GTM-based QSAR models [3], cf. Chapter 16. The ISIDA fragments were not optimized and set as sequences of atoms and bonds of length 2 to 8, generated by ISIDA Fragmentor v. 2013 [20, 99]. For all GTM methods, we rejected descriptors with more than 90% null values. The 186 MOE 2D descriptors defining the "property space" include structural, topological and physicochemical properties, and were generated by MOE v. 2011 [21].

11.4.2 Methods

We compared 25 different models for each property (for conventional and Stargate GTM), obtained by varying the w and λ parameters and keeping the M and K parameter fixed at 25 and 400, respectively. The set of possible w values was [0.25, 0.5, 1, 1.5, 2], the set of λ values [0.01, 0.1, 1, 10, 100]. The "best" out of the 25 models was selected by computing the 3-fold cross-validated Q^2, which assessed the internal performance of the model for the training set. For conventional GTM and S-GTM models, we used the 3-fold cross-validated Q^2 to select the best model parameters. S-GTM models have a multidimensional activity space; therefore, we selected models which had the highest number of properties with $Q^2 > 0.50$. The external test set was then projected onto this "best" model, and an activity prediction was made for each test set molecule. Computing the determination coefficient R^2 assessed the final performance of the model. Thus, at the end of the process, we had one R^2 value per property. We used the GTM k-NN approach to make the predictions for conventional GTM, and the S-GTM mapping from one space to another as a prediction method for S-GTM.

11.4.3 Results and Discussion

For each of the 186 MOE properties, we found an optimal set of parameters [λ, w] for conventional GTM, for all 4 datasets. For Stargate GTM, only one set of parameters was selected for each dataset (cf. Table 11.3). New models were built on the training sets

Table 11.3: Selected parameters [λ, w] for 3-fold cross-validated Stargate GTM regression models, with the corresponding percentage of properties with a determination coefficient R^2 greater than 0.5, the average determination coefficient R^2 and root mean square error $RMSE$ for all properties, and the associated standard deviations.

Dataset	**[λ, w]**	**$\%R^2$ >0.5**	**Average R^2**	**Average RMSE**
ChEMBL	[0.01, 2]	0.54%	0.25 ± 0.20	0.13 ± 0.05
LogS	[0.01, 0.5]	35.7%	0.36 ± 0.23	0.12 ± 0.05
Lu	[100, 1.5]	48.3%	0.36 ± 0.28	0.14 ± 0.05
Thrombin	[100, 0.5]	77.9%	0.54 ± 0.28	0.12 ± 0.05

with these parameters, and the external test sets were projected onto these optimized

maps. Determination coefficients (R^2) were obtained for all properties, for both Stargate and conventional GTM. The R^2 plots as a function of the MOE property are given in Figure 11.8. They show that Stargate GTM generally performs as well as conventional GTM (slightly better for the Thrombin dataset, worse for the LogS dataset).

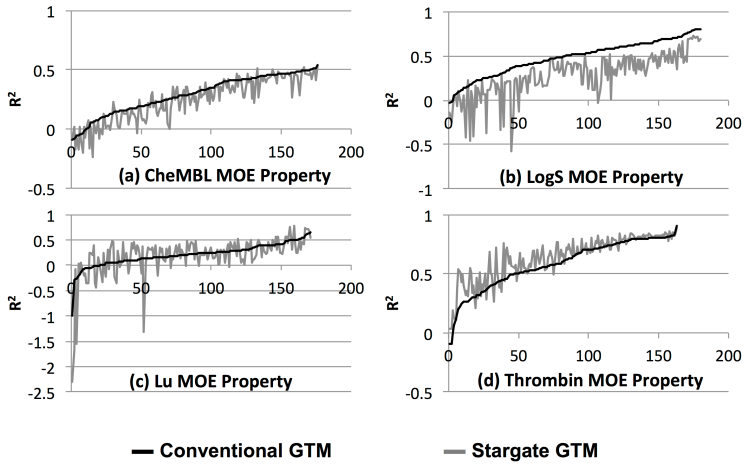


Figure 11.8: Performance of conventional (black) and Stargate GTM (grey) regression models predicting each MOE property, in terms of the determination coefficient (R^2) computed for the external test. The x-axis represents MOE properties identifiers, and data points were sorted by increasing conventional GTM R^2.

11.4.4 Conclusion

In this section, we compared the performance of GTM and S-GTM for predicting MOE properties using ISIDA descriptors. In this experiment, there was no compelling evidence that the S-GTM model based on an 186-dimensional activity space was any better than the conventional GTM models. This contradicts our previous experiments with 8-dimensional or 1-dimensional activity spaces, where S-GTM achieved better performances. This may indicate that, for S-GTM, only a few useful activities should be chosen for the activity space, *i.e.*, a small and relevant activity space should be used for S-GTM regression models.

11.5 Going Further: Impact of Weight Factors

Weight factors w^{Space1} and w^{Space2} ranging from 0 to 1 are attributed to each space to compute the joint responsibilities; one weight determines the other since $w^{\text{Space1}} = 1 - w^{\text{Space2}}$. We investigated the effect of these parameters on the joint responsibilities by systematically varying the weight of the descriptor space form 0% to 100% by steps of 10%. We used the Lu data set for this experiment, and built activity landscapes by

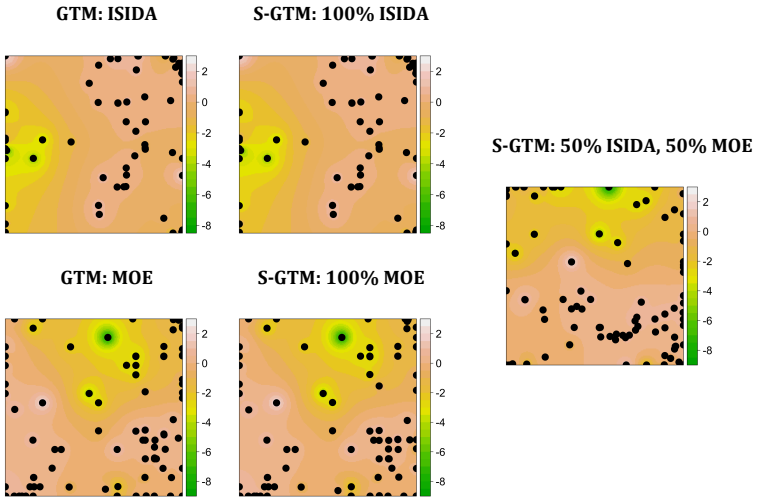


Figure 11.9: S-GTM LogS landscapes for the Lu dataset obtained by varying the weight given to ISIDA and MOE descriptors, compared to the classical GTM landscapes, built with only MOE or ISIDA descriptors.

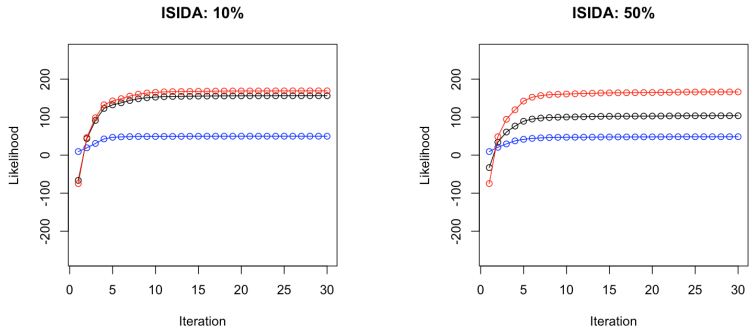


Figure 11.10: Evolution of the S-GTM log likelihood as a function of the number of iterations. The blue line represents the 100% ISIDA log likelihood, the red line the 100% MOE log likelihood, and the black line the S-GTM log likelihood (ISIDA and MOE) with different weights for ISIDA and MOE spaces: (a) 10% ISIDA, 90% MOE, and (b) 50% ISIDA, 50% MOE.

mapping a MOE property (LogS) onto Stargate GTM and conventional GTM models. The landscapes are shown in Figure 11.9; since we wanted to investigate the effect of weights on the joint responsibilities $\mathbf{R}$, we used $\mathbf{R}$ to compute the S-GTM landscapes as well as the mean positions of data points, instead of using $\mathbf{R}^{\text{Space1}}$ or $\mathbf{R}^{\text{Space2}}$. The Stargate and conventional GTM models were built with the parameters selected for Stargate GTM $[K = 400, M = 25, \lambda = 100, w = 1.5]$. We mapped the LogS MOE property to see the evolution of the map's shape. As could be expected, the 100% ISIDA Stargate model is almost identical to the conventional ISIDA map, and the 100% MOE Stargate model almost identical to the conventional MOE map. The 50% ISIDA, 50% MOE Stargate map is a mix of both ISIDA and MOE maps.

In Figure 11.10, we show the evolution of the log likelihood as a function of the number of iterations for two different sets of weights: the S-GTM log likelihood will be closer to the 100% ISIDA log likelihood if the weight for ISIDA is larger. Figures 11.9 and 11.10 show how the weights can have an impact on the model; a user might want to give more importance to one space or another for visualization, and it may be necessary to optimize the weights to obtain a better prediction performance.

11.6 Going Further: Impact of Property Correlation

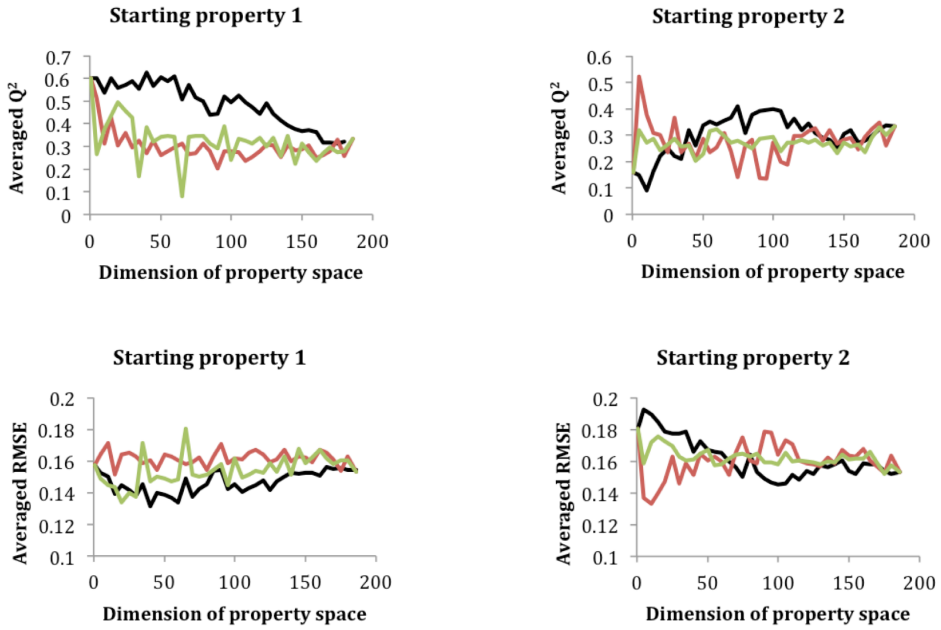


Figure 11.11: Average performance (for all properties in the model) of Stargate models for the Lu dataset when adding new properties to be predicted, sorted by decreasing correlation to a starting property (black line), or randomly (colored lines).

In order to investigate the impact of property correlation on prediction performances within the S-GTM activity space, we applied a specific procedure to all four datasets. A MOE property P_{start} was chosen at random and the other MOE properties were sorted by decreasing correlation to P_{start}. A series of Stargate models were built (with fixed parameters $w = 1$, $l = 1$, $M = 25$, $K = 225$), beginning with a model with only P_{start}, then including other properties into the model five by five by decreasing correlation. A control experiment was conducted, starting from P_{start} and including more and more properties, but this time in a random order. The control experiment was performed two times with a different random order. We could therefore compare the "ordered" case to two "random" cases, and see if using correlated properties was having an effect on the performance of S-GTM models. The procedure (one ordered

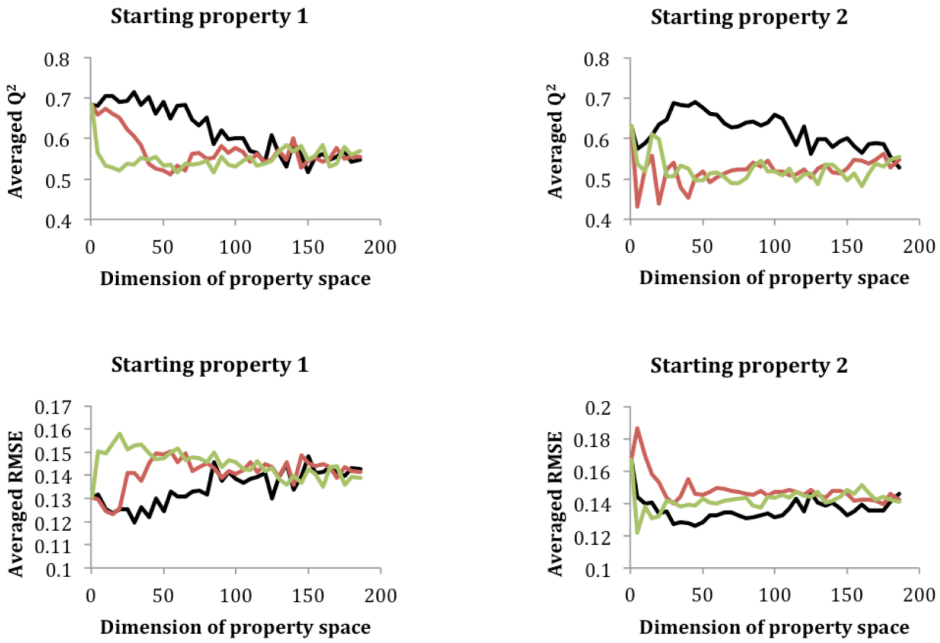


Figure 11.12: Average performance (for all properties in the model) of Stargate models for the Thrombin dataset when adding new properties to be predicted, sorted by decreasing correlation to a starting property (black line), or randomly (colored lines).

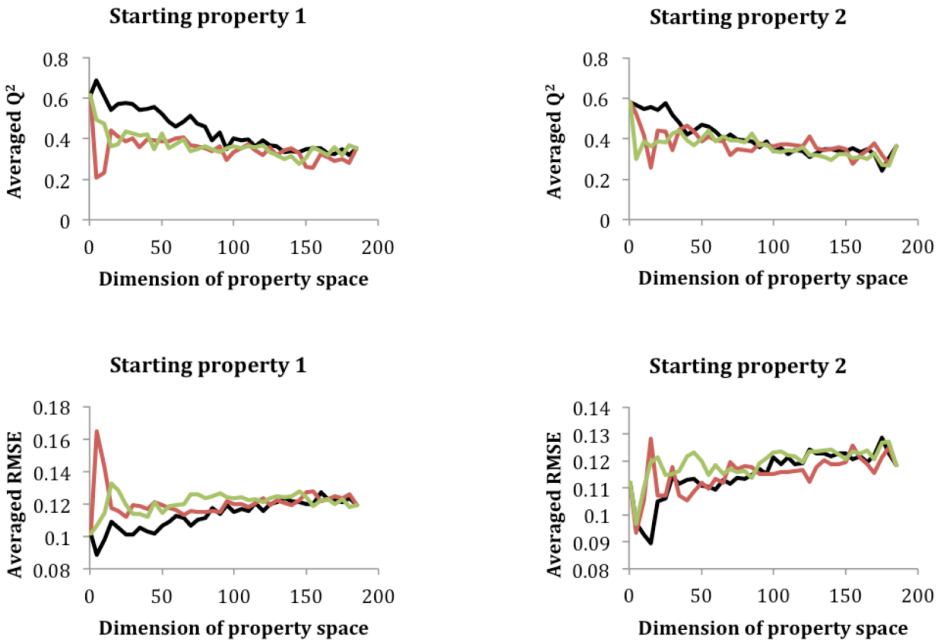


Figure 11.13: Average performance (for all properties in the model) of Stargate models for the LogS dataset when adding new properties to be predicted, sorted by decreasing correlation to a starting property (black line), or randomly (colored lines).

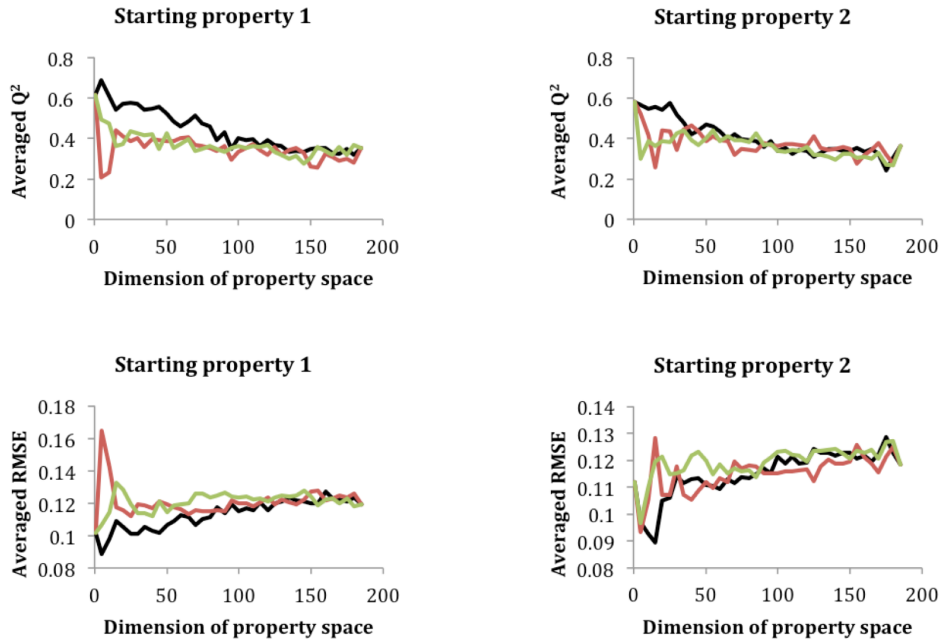


Figure 11.14: Average performance (for all properties in the model) of Stargate models for the ChEMBL dataset when adding new properties to be predicted, sorted by decreasing correlation to a starting property (black line), or randomly (colored lines).

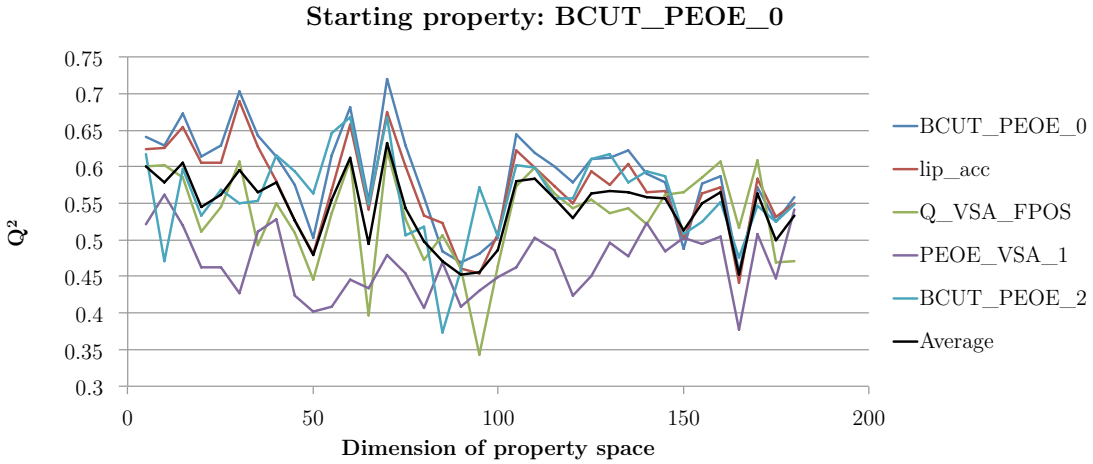


Figure 11.15: Q^2 of Stargate models for the Lu dataset for 5 individual correlated properties shown as colored lines (P_{start} and its 4 most correlated properties), and averaged Q^2 for the 5 properties (black line), as a function of the number of properties included in the Stargate model, sorted by decreasing correlation to the starting property (BCUT_PEOE_0).

experiment, two random control experiments) was repeated two times with two different starting properties $P_{start}(1)$ and $P_{start}(2)$. We can only observe a small effect (see Figures 11.11 to 11.14) demonstrating that by adding less correlated properties, the average ability of S-GTM models to predict them decreases. However, by only selecting

some individual properties, as in Figure 11.15, where we selected one random property and its 4 closest neighbors in terms of correlation, the effect is harder to see. It is interesting to observe in Figure 11.15 that the performance seems to follow the same trend for these correlated properties: for example, there is a low performance for all properties when the dimensionality of the property space ranges from 75 to 100.

11.7 Conclusion

We introduced a new multispace visualization and regression method coined Stargate GTM. With S-GTM, it is possible to jump from one vector space to another, by passing through a 2D dimensionality reduction (a 2D "gate"). The user can tune the weight assigned to each space. This method can be used in chemistry to travel from a descriptor space to a property space and vice versa, to predict a whole activity profile based on structural information, and to delimit regions populated by structures corresponding to an activity profile. We showed that the performance of Stargate GTM for regression can be at least the same or better than conventional GTM, depending on the relevance of individual activities in the S-GTM activity space.

Chapter 12

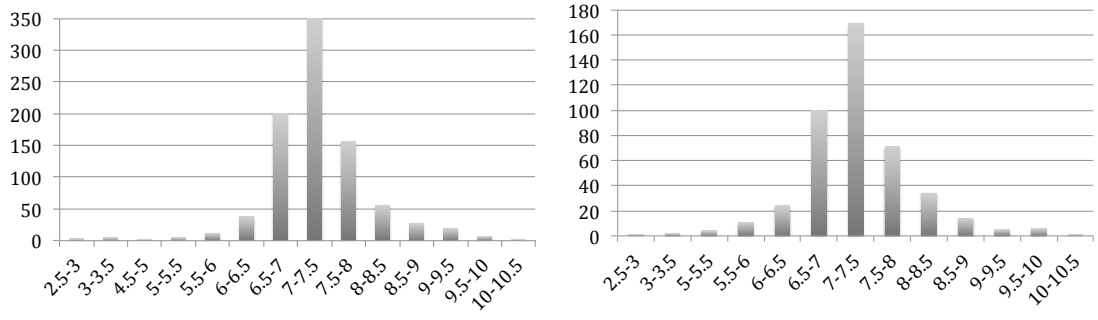
Inverse QSAR

12.1 Introduction

In this chapter, we introduce some inverse QSAR methodologies based on the conventional GTM and the "Stargate" algorithm introduced in Chapter 11. Inverse QSAR is usually defined as the process of generating new structures based on activity information. This can be done in two steps, by first estimating the descriptors corresponding to a specific activity profile and then by generating molecular graphs from descriptors [106, 107]. Our definition of inverse QSAR is slightly different; we estimate descriptors based on activity information and retrieve known structures corresponding to an activity query, but we do not generate new chemical structures. We developed two types of GTM-based and S-GTM-based methods: (1) methods generating approximate descriptor values or "structure prototypes", which retrieve so-called "best matching structures" (BMSs) from a database using a k-NN approach; (2) an approach similar to virtual screening, which employs a scoring function to retrieve the BMSs corresponding to an activity query; the BMSs are then equivalent to "hits" in virtual screening.

12.2 Data and Descriptors

For all our inverse QSAR experiments, we used the dataset of 1325 ChEMBL molecules from the foregoing Chapter 11 dedicated to S-GTM, described by the same ISIDA descriptors, and annotated by the affinity for Serotonin 1a. We used only one activity (the affinity for Serotonin 1a) to find new structures, but it could have been an entire activity profile, as we demonstrated in Chapter 11. The dataset was split into a training set of 883 compounds and a test set of 442 compounds. The activity of training and test set compounds is shown in Figure 12.1.



(a) Training set.

(b) Test set.

Figure 12.1: Number of training and test set compounds as a function of the affinity for Serotonin 1a.

12.3 Inverse QSAR with Conventional GTM

For inverse QSAR, the input data is an activity (or multiple activities); our goal is to find chemical structures that also exhibit this activity. A way to do this with a simple GTM model is to project the activity (or multiple activities) onto the map and obtain an activity landscape (or multiple activity landscapes). On the landscape, a landscape activity value $\bar{a}_k$ is assigned to each node $\mathbf{x}_k$; the coordinates of a node can be considered as a 2D latent prototype, *i.e.*, a type of average molecule at a specific location on the map.

We can select activity landscapes node(s) with a desired activity. Then, two possibilities might arise. We may simply map molecules from the database and see which molecules lie near this 2-dimensional node (latent prototype), or we may generate a D-dimensional structure prototype made up of estimated descriptor values (D-dimensional prototype). The latent prototype can be used to look for nearest neighbors on the 2D map (latent space), and D-dimensional prototypes for nearest neighbors in the initial space (D-dimensional descriptor space). There are at least 2 ways of generating the D-dimensional prototype:

1. Manifold prototype: after selecting the 2-dimensional node with the desired activity, we may simply retrieve its D-dimensional coordinates on the manifold to obtain a D-dimensional prototype, since all manifold nodes have 2-dimensional coordinates and D-dimensional coordinates. The manifold prototype is the equivalent of the latent prototype in the D-dimensional space.

2. Landscape prototype: after selecting the 2-dimensional node with the desired activity, we map all D descriptors onto the 2-dimensional map and get D landscapes. From these D landscapes, we can reconstitute a prototype for the node we are

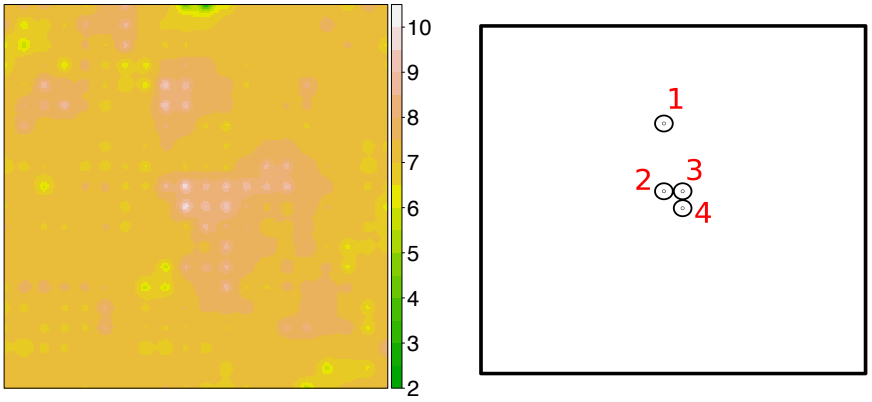
interested in.

A structure prototype is generated by a GTM model and benefits from the information provided by all training set compounds; therefore, the GTM approach is quite different from simply computing similarities between individual molecules in the initial descriptor space. Moreover, we might identify structure prototypes with the same activity in different regions of the chemical space.

In the following sections, we will study 3 inverse QSAR methods applied to conventional GTM:

1. 2-dimensional latent prototype (requires to map a database onto the model).
2. *D*-dimensional manifold prototype.
3. *D*-dimensional landscape prototype.

These methods depend upon the construction of an activity landscape or several affinity landscapes for multiple activities. Here, we only have one activity to model: the affinity for Serotonin 1a. The corresponding landscape is shown in Figure 12.2a.



(a) S1a landscape. **(b)** S1a regions of interest.

Figure 12.2: (a) Landscape of Serotonin 1a affinity, (b) high Serotonin 1a affinity nodes (> 8.9).

12.3.1 Latent Prototype

The simplest GTM-based inverse QSAR approach is the latent prototype, based on finding database molecules near a GTM node corresponding to the activity query. The 2-dimensional latent space is characterized by 2 latent variables, which can be considered as hidden structural features, and our latent structure prototype is made up of the 2D

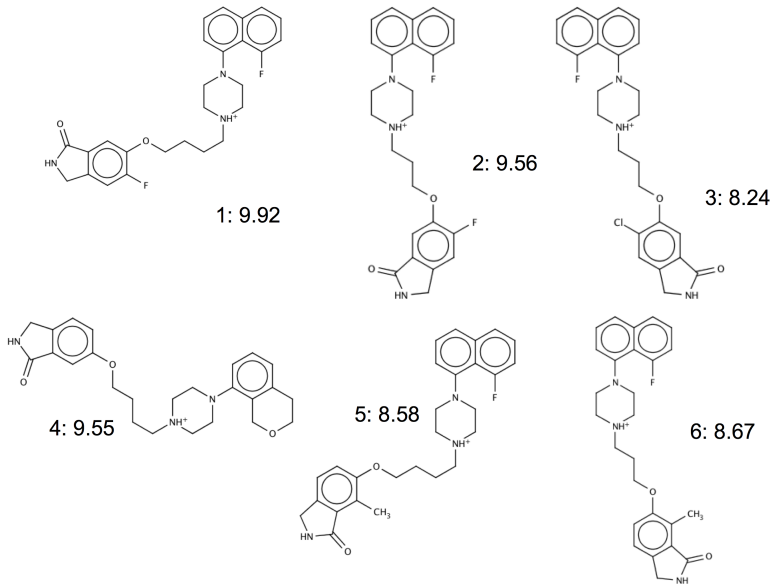


Figure 12.3: Structures (with their experimental affinities for Serotonin 1a) retrieved within region 1 in Figure 12.2b with a predicted affinity > 8.9.

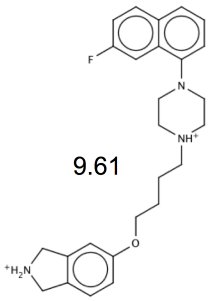


Figure 12.4: Structure (with experimental affinity for Serotonin 1a) retrieved within region 2 in Figure 12.2b with a predicted affinity > 8.9.

coordinates of the GTM node of interest. It can be seen as an average molecule at a given position on the map. The workflow is the following:

1. Train a GTM model with the training set.
2. Build an activity landscape using the training set.
3. Project the test set (= the database) onto the GTM model.
4. Look for the 2D nearest neighbors of the latent prototype (GTM node) with a specific activity.
5. Apply a density-based applicability domain: only sufficiently populated nodes can become prototypes.

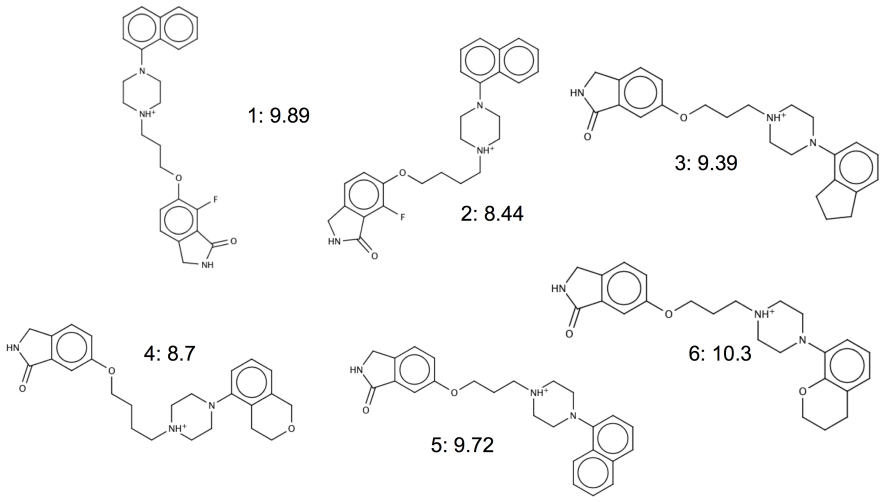


Figure 12.5: Structures (with their experimental affinities for Serotonin 1a) retrieved within region 3 in Figure 12.2b with a predicted affinity > 8.9.

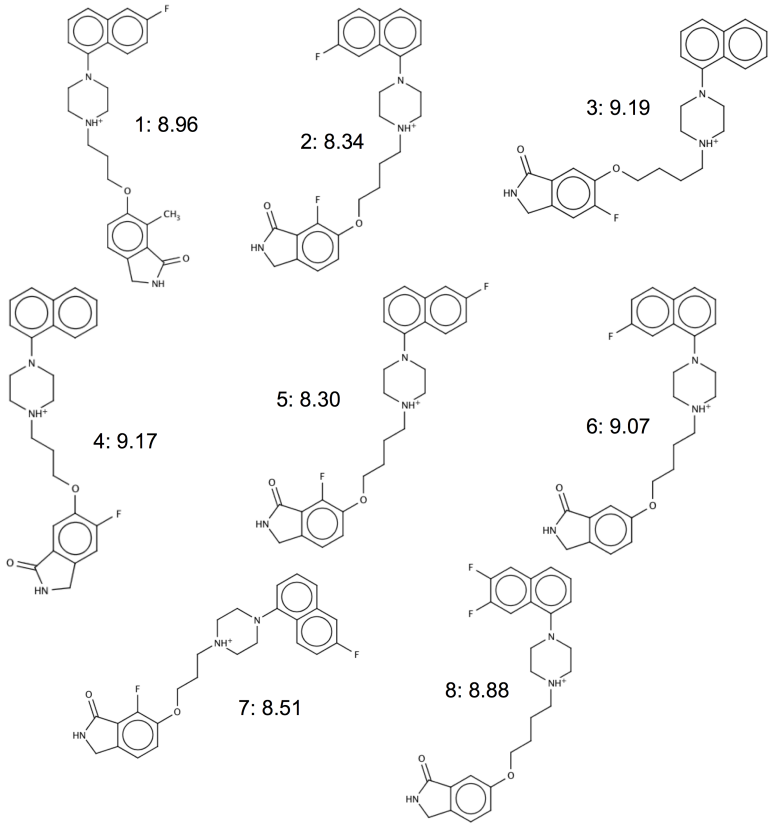
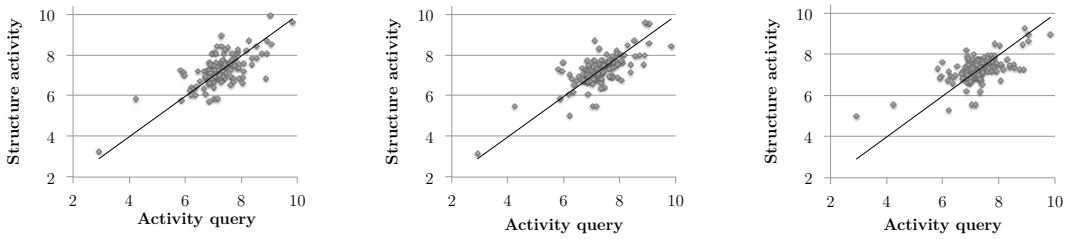


Figure 12.6: Structures (with their experimental affinities for Serotonin 1a) retrieved within region 4 in Figure 12.2b with a predicted affinity > 8.9.

For this method, the activity queries are the activity landscape values. An activity is assigned to each node (latent prototype); therefore, we should be able to retrieve structures with a similar activity near the prototype. There are two ways to do this: draw a region around each prototype and find the molecules mapped within this region (we use confidence ellipses for this; cf. Figure 12.2b), or simply compute the distance between nodes and compounds in the latent space and retrieve the nearest neighbors. Applying the applicability domain (AD) is a necessary step here, since nodes that are not well populated by the training set cannot be "trusted". The main advantage of using this approach is the easy identification of zones sometimes far away from each other in the descriptor space but with a similar activity. For example, regions of high activity (affinity > 8.9) are shown in Figure 12.2b. Several compounds were found near the four prototypes with affinity > 8.9 and are shown in Figures 12.3, 12.4, 12.5 and 12.6, corresponding to regions 1, 2, 3 and 4 in Figure 12.2b, respectively. There were 23 retrieved structures, amongst which all 12 compounds with an experimental activity > 8.9 were recovered; the other 11 retrieved compounds had similar structural characteristics but slightly lower activities. We plotted the experimental activity of best matching struc-



(a) 1 BMS. **(b)** 3 BMSs. **(c)** 10 BMSs.

Figure 12.7: Average activity of best matching structures (BMSs) as a function of the activity query in an inverse QSAR task. BMSs are the nearest neighbors of the structure prototype on a 2D GTM map, and the activity query is the activity assigned to the structure prototype.

tures (BMSs) in the test set against the activity of latent prototypes (activity queries) in Figure 12.7, to see if the method could identify compounds corresponding to a specific activity query. For the problem at hand, the test set was quite small (442 compounds), and the number of nodes high (400) and trying to find too many compounds per query would ineluctably lead to failure; therefore, we did not consider more than 10 BMSs. We used the correlation and determination coefficients (Table 12.1) to systematically compare the activity query (the activity of each prototype) to the average activity of best matching structures. This method shows promising results and has the advantage of being simple, fast, and easily interpretable. Instead of a single activity, an activity profile can also be used as a query by constructing multiple activity landscapes, finding

regions of interest (ROIs, cf. Chapter 5) corresponding to criteria applicable to each landscape, and selecting prototypes within these regions of interest. One of the drawbacks of this method is that we cannot use any activity query we want and look for the corresponding molecules in the database: we have to find the landscape activity closest to the desired activity and only then look for the best matching structures.

12.3.2 Manifold Prototype

The second inverse QSAR approach that we tested was the manifold prototype: instead of using the 2D coordinates of nodes in the latent space, we selected their coordinates on the manifold in the D-dimensional descriptor space. Each D-dimensional manifold node $\mathbf{y}_k$ will be considered a structural prototype, with a predicted activity value from the affinity landscape; a structure with descriptors similar to that prototype should have an activity similar to that of the prototype. The workflow was the following:

1. Train a simple GTM model with the training set.

2. Build an activity landscape using the training set activity.

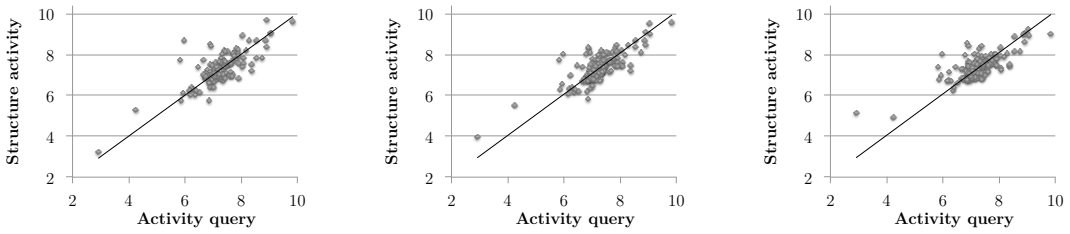
3. Look for a node with a specific activity and retrieve its coordinates in D-dimensional space: this will be the structural prototype.

4. Compute distances in descriptor space between the prototype and the database molecules, and retrieve the best matching structures, which should have an activity similar to that of the prototype.

5. Apply a density-based applicability domain: only sufficiently populated nodes are considered.

We get the prototypes by simply retrieving the D-dimensional coordinates of nodes of interest on the manifold. We plotted the activity of the best matching structures in the test set against the activity of prototypes (Figure 12.8), to see if the method could allow us to systematically find compounds corresponding to an activity query. The corresponding correlation and determination coefficients can be found in Table 12.1.

This method yields results close to those of the 2-dimensional approach, although the manifold prototype is based on a different idea. The advantage of this method is that we work with prototypes based on original descriptors predictions. However, these prototypes do not correspond to any actual molecular structure; they are D-dimensional vectors that should be similar to descriptor vectors related to actual molecular structures.



(a) 1 BMS. **(b)** 3 BMSs. **(c)** 10 BMSs.

Figure 12.8: Average activity of best matching structures (BMS) as a function of the activity query in an inverse QSAR task. BMSs are the nearest neighbors of the manifold-based structure prototype in D-dimensional space, and the activity query is the activity assigned to the structure prototype.

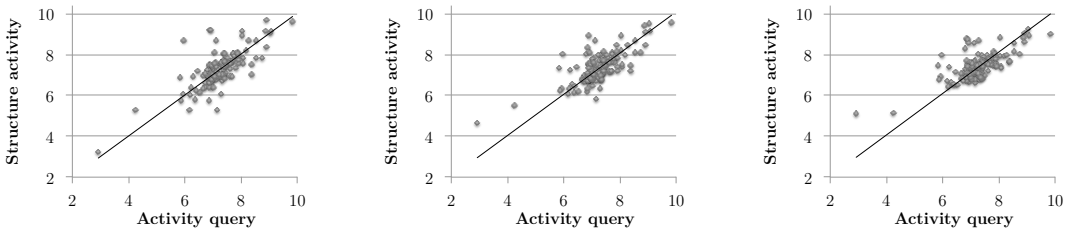
12.3.3 Landscape Prototype

The last method that we explored for inverse QSAR with conventional GTM was the landscape prototype. The process is quite similar to the manifold prototype: we want to generate a D-dimensional prototype and use it to find neighboring structures in D-dimensional space. This time, instead of using manifold nodes as prototypes, we used multiple descriptor landscapes (cf. Chapter 5) to predict descriptor values for each GTM node. We designed the following workflow:

1. Train a simple GTM model with the training set.
2. Build an activity landscape using the training set.
3. Build descriptor landscapes for all D descriptors. Each node will therefore be associated with a predicted D-dimensional vector built from these landscapes.
4. Compute distances in descriptor space between the prototype and the database molecules, and retrieve the best matching structures, which should have an activity similar to that of the prototype.
5. Apply a density-based applicability domain: only sufficiently populated nodes are considered.

We plotted the activity of the best matching structures in the test set against the predicted activity of prototypes (Figure 12.9), to see if the method would allow us to systematically find compounds within our database matching the activity query. The corresponding correlation and determination coefficients can be found in Table 12.1.

This method is more time-consuming than the latent prototype or manifold prototype approaches; if molecular structures are described by D descriptors in the database, D landscapes have to be built to construct the landscape prototypes.



(a) 1 BMS. **(b)** 3 BMSs. **(c)** 10 BMSs.

Figure 12.9: Average activity of best matching structures (BMSs) as a function of the activity query in an inverse QSAR task. BMSs are the nearest neighbors of the landscape-based structure prototype in the D-dimensional space, and the activity query is the activity assigned to the structure prototype.

Table 12.1: R^2 and Pearson correlation coefficients for the 3 inverse QSAR method, estimated by comparing the prototype activity (= activity query) to the experimental activity of 1, 3 or 10 best matching structures (BMSs).

Pearson correlation	**1 BMS**	**3 BMSs**	**10 BMSs**
GTM: latent prototype	0.71	0.73	0.63
GTM: landscape prototype	0.78	0.78	0.75
GTM: manifold prototype	0.75	0.73	0.70
R^2	**1 BMS**	**3 BMSs**	**10 BMSs**
GTM: latent prototype	0.43	0.49	0.37
GTM: landscape prototype	0.55	0.56	0.53
GTM: manifold prototype	0.42	0.46	0.43

12.3.4 Conclusion

The performance of our GTM-based inverse QSAR models are given in Table 12.1. The determination and correlation coefficients were computed by comparing the landscape activities (= queries) and the activities of the best matching compounds found with the different methods. The "landscape prototype" approach seems to be the best for conventional GTM.

12.4 Inverse QSAR with S-GTM

With S-GTM, we can travel from the descriptor space to property space and vice versa, by going through a latent 2-dimensional space. For inverse QSAR, the input data is an activity or an entire activity profile; the goal is to find chemical structures possessing a similar profile. The conventional GTM methods shown in the foregoing section are also applicable to S-GTM; we will also explore other ways to perform inverse QSAR with

S-GTM.

12.4.1 Multimodality Problem

We will first expose a problem already highlighted in the chapter dedicated to Stargate GTM (Chapter 11). If a structure travels to activity space, its distribution at arrival will be unimodal: a structure maps to a single activity profile. However, when an activity travels to the descriptor space, the distribution at arrival is usually multimodal, since a single activity profile maps to multiple structural regions; this is the multimodality problem. This phenomenon is already encountered in activity landscapes: different regions may have similar activity values without being close to each other.

Therefore, in the case of an activity traveling to descriptor space (inverse QSAR) through S-GTM, we cannot summarize the distribution in the 2-dimensional latent space by the mean position on a 2-dimensional map. The most obvious (and wrong) way to construct a "prototype" vector would be the following:

1. Train an S-GTM model.
2. Project a desired activity vector onto the 2-dimensional S-GTM model (the Stargate door).
3. Map the MEAN 2-dimensional position to descriptor space and obtain a D-dimensional prototype.

However, this method would not be efficient, especially if the dimension of the activity space were low. Why? - Because of the multimodality problem. More specifically, when we project the desired activity vector onto the 2-dimensional S-GTM (step 2 in the workflow), we obtain a probability distribution as shown in Figure 12.10. The

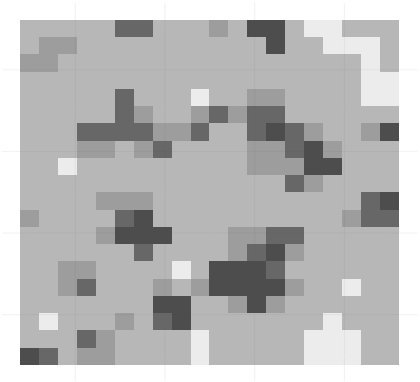


Figure 12.10: 2-dimensional probability distribution obtained when mapping an activity to the 2-dimensional S-GTM map.

activity is not just located in one region. The darker regions represent zones where the probability of finding the property of interest is higher. We can see that this distribution has multiple modes. If we applied the previous workflow, we would just take the mean point of this distribution, and then map it to descriptor space. This would be a good approach if the distribution were unimodal. Since it is not the case, this approach was discarded.

12.4.2 Multiple Prototypes

A way to account for the multimodality of activities projected onto the S-GTM 2D map is to use multiple prototypes for one compound, corresponding to the multiple modes of the activity distribution. We adopted the following workflow to investigate this approach:

1. Train an S-GTM model.
2. Project 147 artificial "Serotonin 1a" activity samples, ranging from 3 to 10.3 by steps of 0.05 and obtain the matrix of responsibilities $\mathbf{R}$.
3. For each activity a, select the n nodes $\{\mathbf{x}_k\}$ with the highest probability, for example, $R_{ka} > 0.001$. The n selected nodes could be used to select n prototypes corresponding to manifold nodes in descriptor space (manifold prototypes) or in 2D latent space (latent prototypes). If the activity distribution is monomodal, there is one structure prototype per activity. If it is multimodal, there are n structure prototypes per activity.

The problem with this method is that the number of modes for one molecule (and therefore the number of prototypes) is sometimes considerable and these prototypes might become meaningless. To illustrate this, we plotted the number of nodes with $R_{ka} > 0.001$ for all tested activity values (Figure 12.11).

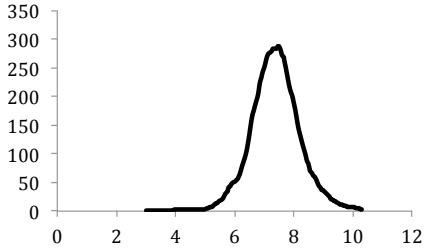


Figure 12.11: Number of nodes $\{\mathbf{x}_k\}$ with $R_{ka} > 0.001$ as a function of the activity landscape value a.

The number of selected nodes ($R_{ka} > 0.001$) as a function of the activity value gives a bell-shaped curve: the more extreme the activity, the fewer the nodes. This means that

extreme (< 6 or > 9) activity values are more likely to be more localized when projected onto the 2-dimensional map. This is linked to the number of training set compounds in each activity range, as can be seen in Figure 12.1: only a few training set compounds have extreme affinities for Serotonin 1a. Therefore, the number of prototypes that should be tested is actually quite low for very high or very low activities, and quite large if the values are in the middle range. For example, for an S1a affinity = 7.3, we retrieve 284 nodes with probability > 0.001, which is more than half of the total of number of nodes on the map and makes this methodology inefficient. This approach, however, can be applied to retrieve structures when the selected structural regions are just a few. This situation arises when we want to identify actives and separate them from inactives. The actives often have very specific features and are localized within specific regions of the map; this was already shown with the latent prototype approach when we selected only four nodes with a Serotonin 1a activity > 8.9 (Figure 12.2).

12.4.3 Discarded Method: Weighted Landscape Prototype

Another way to create D-dimensional structural prototypes is to build descriptor landscapes based on the training set, as we did previously for conventional GTM. However, this time, instead of looking for structures near a node on the 2-dimensional map, all responsibilities $\mathbf{R}$ (posterior probabilities in latent space for the projected Serotonin 1a activities, cf. Figure 12.10 for the distribution of an activity) are taken into account to predict each descriptor value. In other words, we try to approximate a unique set of descriptors, a structural prototype, for a given activity query. The workflow is the following:

1. Use a training test for training an S-GTM model.

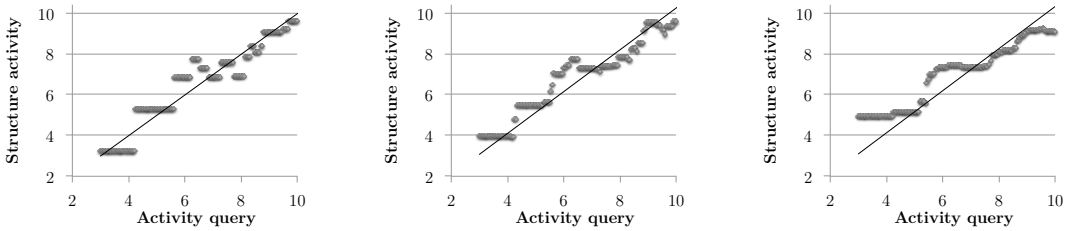
2. Map each descriptor onto the map and obtain 144 landscapes.

3. Project 147 artificial "Serotonin 1a" activities, ranging from 3 to 10.3 by steps of 0.05.

4. Using the responsibilities $\mathbf{R}$ for the projected Serotonin 1a activities and the D descriptor landscapes $\{\bar{\mathbf{a}}(1), \bar{\mathbf{a}}(2), ..., \bar{\mathbf{a}}(D)\}$, compute the 144 average descriptor values of each artificial activity a, and obtain a 144-dimensional "prototype" vector for each sample activity. The estimated descriptor value $\hat{T}_{ad}$ for an activity query a is computed as following:

$$\hat{T}_{ad} = \frac{\sum_k R_{ka} \bar{a}_k(d)}{\sum_k R_{ka}} \tag{12.1}$$

This method takes into account the whole probability distribution in 2D for a projected activity. It also produces one vector prototype per compound, which makes it easy to use. However, unlike the conventional GTM landscape prototype, this weighted landscape prototype is just an average of diverse descriptor values and is not representative of the fact that many structures correspond to a single activity query. Therefore, it does not account for multimodality in a broader sense, and is not a good approximation of possible descriptor values. Nevertheless, we applied this method to search for matching structures corresponding to activity queries. We plotted the average activity of the best matching structures against the requested activity (Figure 12.12).



(a) 1 BMS. **(b)** 3 BMSs. **(c)** 10 BMSs.

Figure 12.12: Average activity of best matching structures (BMSs) as a function of the activity query in an inverse QSAR task. BMSs are the nearest neighbors of the weighted landscape-based structure prototype in the D-dimensional space, and the activity query is the activity assigned to the structure prototype.

Just looking at Figure 12.12, we can see that this approach works, although it was not the expected result. The correlation coefficients between the requested and retrieve activities can be found in Table 12.2. This method seems to generate an "average vector" which is able to find matching structures. However, we did not retain this approach as a valid method, since the weighted landscapes did not correspond to any real structural approximation.

12.4.4 Conditional Log Likelihood Approach

The conditional log likelihood approach consists in using a conditional log likelihood as a scoring function for molecules in the test set. Molecules with the highest score for a given activity are selected. We applied the following workflow:

1. Use a training test to train an S-GTM model.

2. Project 147 artificial Serotonin 1a activities, ranging from 3 to 10.3 by steps of 0.05. The responsibilities $\mathbf{R}_{ka}$ of each node $\mathbf{x}_k$ for the activity query a will be used in (4).

3. Project compounds of the test set onto the map. The PDF $p(\mathbf{t}_q|\mathbf{x}_k, \mathbf{W}, b)$ for each compound $\mathbf{t}_q$ and node $\mathbf{x}_k$ will be used in (4).

4. Using $\mathbf{R}_{ka}$ computed in (2) and $p(\mathbf{t}_q|\mathbf{x}_k, \mathbf{W}, b)$ computed in (3), calculate the conditional log likelihood for each test set compound. There will be one conditional log likelihood value for each [test set structure q, artificial activity a] combination.

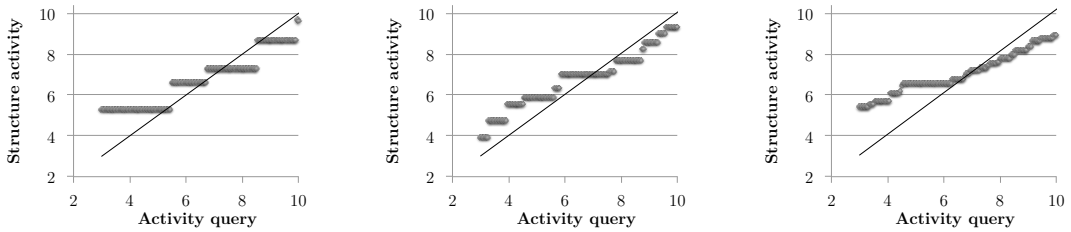
5. For each artificial activity, take n best matching structures or BMSs in terms of conditional log likelihood (n = 1, 3, and 10).

6. Plot the average activity of the best matching structures found in (5) against the artificial activity queries.

The formula for the conditional log likelihood $\mathcal{L}_{qa}$ assigned to the qth structure and ath activity profile is the following:

$$\mathcal{L}_{qa} = \ln\left(\frac{1}{K}\sum_{k}^{K}(R_{ka} \times p(\mathbf{t}_q|\mathbf{x}_k, \mathbf{W}, \beta))\right) \tag{12.2}$$

where R_{ka} is the responsibility of node $\mathbf{x}_k$ for the requested activity a, $p(\mathbf{t}_q|\mathbf{x}_k, \mathbf{W}, \beta)$ the probability density for each test compound $\mathbf{t}_q$ given a node $\mathbf{x}_k$, and K the number of nodes.



(a) 1 BMS. **(b)** 3 BMSs. **(c)** 10 BMSs.

Figure 12.13: Average activity of best matching structures (BMSs) as a function of the activity query in an inverse QSAR task; BMSs are the structures with the highest conditional log likelihood score.

In Figure 12.13, we show the results featuring the average activity of best matching structures (BMSs) found using the conditional log likelihood as a scoring function as a function of the requested activity. In Table 12.2, we also show the corresponding correlation and determination coefficients for this method (S-GTM: likelihood) as well as the previous method (S-GTM: weighted landscape prototype). Both methods perform well; but the likelihood method is to be preferred, since the landscapes approach does not account for multimodality and is not a conceptually valid approach..

Table 12.2: Pearson correlation coefficient and determination coefficient R^2 comparing activity queries (affinity values ranging from 3 to 10.3 by steps of 0.05), to the experimental activity of 1, 3 or 10 best matching structures or BMSs using S-GTM-based inverse QSAR methods.

Pearson correlation	**1 BMS**	**3 BMSs**	**10 BMSs**
S-GTM: weighted landscape prototype	0.97	0.97	0.97
S-GTM: manifold prototype	0.96	0.98	0.98
R^2	**1 BMS**	**3 BMSs**	**10 BMSs**
S-GTM: weighted landscape	0.84	0.91	0.84
S-GTM: manifold prototype	0.82	0.86	0.68

12.5 Comparing GTM and S-GTM approaches

Table 12.3: R^2 for 4 inverse QSAR methods, estimated by comparing activity queries (sampled from the landscape) to the experimental activity of 1, 3 or 10 best matching structures or BMSs.

	1 BMS	**3 BMSs**	**10 BMSs**
S-GTM: likelihood score	0.63	0.70	0.76
GTM: latent prototype	0.43	0.49	0.37
GTM: landscape prototype	0.55	0.56	0.53
GTM: manifold prototype	0.42	0.46	0.73

Table 12.4: R^2 for 4 inverse QSAR methods, estimated by comparing the queries (affinity values ranging from 3 to 10.3 by steps of 0.05) to the experimental activity of 1, 3 or 10 best matching structures or BMSs.

	1 BMS	**3 BMSs**	**10 BMSs**
S-GTM: likelihood score	0.82	0.86	0.68
GTM: latent prototype	0.76	0.74	0.70
GTM: landscape prototype	0.77	0.75	0.75
GTM: manifold prototype	0.84	0.76	0.73

Finally, we retained four conceptually valid techniques for GTM and S-GTM-based inverse QSAR. 3 methods were described in the section dedicated to conventional GTM and can be applied to both GTM and S-GTM: latent prototypes, manifold prototypes, and landscape prototypes. The fourth method, based on a conditional log likelihood scoring function, is specific to S-GTM. To compare GTM and S-GTM approaches, we did multiple activity queries and compared the ability of all methods to retrieve "correct" structures from the test set. This "test set" can be considered as a database

consisting of molecules with unknown activities. The training set's only purpose was to train the map; to evaluate the method, we only used the test set, which was never "seen" by the model. In Tables 12.3 and 12.4, we summarize the results obtained with conventional GTM and S-GTM by looking for compounds with a Serotonin 1a affinity close to GTM landscape values, or close to 147 "artificial" affinities ranging from 3 to 10.3 by steps of 0.05. Both queries (landscape values, artificial affinities) are different ways of sampling the affinity space; one is representative of the training set population, the other is a systematic search. The performance was evaluated by comparing the queries and the actual test set affinities of retrieved compounds. Among GTM-based methods, the landscapes approach seems to be the most stable, but there is no compelling reason to argue that one method is better than the other. However, to avoid the curse of dimensionality arising when computing distances in a multidimensional space, the latent prototype or conditional log likelihood are the best choices.

12.6 Conclusion

In this chapter, we introduced new inverse QSAR methods based on conventional and Stargate GTM. We selected 4 methods that were able to retrieve structures corresponding to an activity query. The first three methods, applicable to both GTM and S-GTM, are based on 2D latent prototypes, D-dimensional manifold prototypes and landscape prototypes. The fourth approach is specific to S-GTM and employs the conditional log likelihood as a scoring function to retrieve best matching structures for a given activity query. These methods can be easily generalized to handle activity profile queries.

Chapter 13

ISIDA/GTM Software

The ISIDA/GTM software is a chemical space visualization and modelling toolbox based on the GTM algorithm [10, 9] and programmed during this thesis. All GTM variants described in this thesis were implemented into the software, which can therefore be employed for very large datasets (incremental algorithm), handling high dimensions (kernel algorithm KGTM), traveling from descriptor space to property space or vice versa (S-GTM), and for the validation of visual classification and regression models. From a molecular structure data file (SDF), the ISIDA/GTM software generates fragment descriptors (using the external tool ISIDA Fragmentor [99]), builds 2D GTM maps with a link to the chemical structures, generates classification maps and property landscapes, and builds GTM classification and regression models. It is a multipurpose tool providing insight into the underlying structure of a dataset. ISIDA/GTM and the separate command-line tool GTMapTool (Appendix B), will be available from the website `http://infochim.u-strasbg.fr/` after the submission of this thesis.

13.1 Introduction

ISIDA/GTM is composed of a graphical user interface (GUI) GTMap, and a command-line tool, GTMapTool, which can be used separately. Advanced options such as the kernel and incremental algorithms or Stargate GTM are only available in the command-line tool. The GUI uses the GTMapTool program for all GTM-related calculations. GTMapTool takes as input tab-separated descriptors or files in the LIBSVM format [108]. To make our software self-standing, the external command-line tool ISIDA Fragmentor 2013 [99] was integrated to build fragment descriptors from structure data files (SDF format). The user can also load any other descriptor file of his own choice, such as MACCS or MOE descriptors. ISIDA/GTM was entirely programmed in Free Pascal using the Lazarus IDE [109], and was made available for Linux and OS X. The GUI comprises 5 tabs, corresponding to the "natural" workflow:

1. Data and set-up
2. Descriptors
3. Build new GTM
4. QSAR
5. Analysis/visualization
6. Map new molecules onto trained model

The QSAR step is available when there is a property to model, and the user can jump to any step directly. It is possible to go directly to the visualization step if the GTM model was already computed with the command-line tool. All results are saved into specified output directories and can be loaded into the software at any time. The author of this thesis implemented the whole GTMapTool program and ISIDA/GTM interface, but was not involved in the implementation of the Fragmentor program [99] nor of the molecular structure visualizer *MolDraw* programmed by Vitaly P. Solov'ev.

13.2 Data and Set-Up

In the *Data and Set-Up* tab (Figure 13.1), the user defines the output directory and the location of the SDF file. In all tabs, the output directory can be changed. A property can be selected for modeling tasks by loading it from an external file or directly from the SDF file. The software supports class labels as well as continuous activity values.

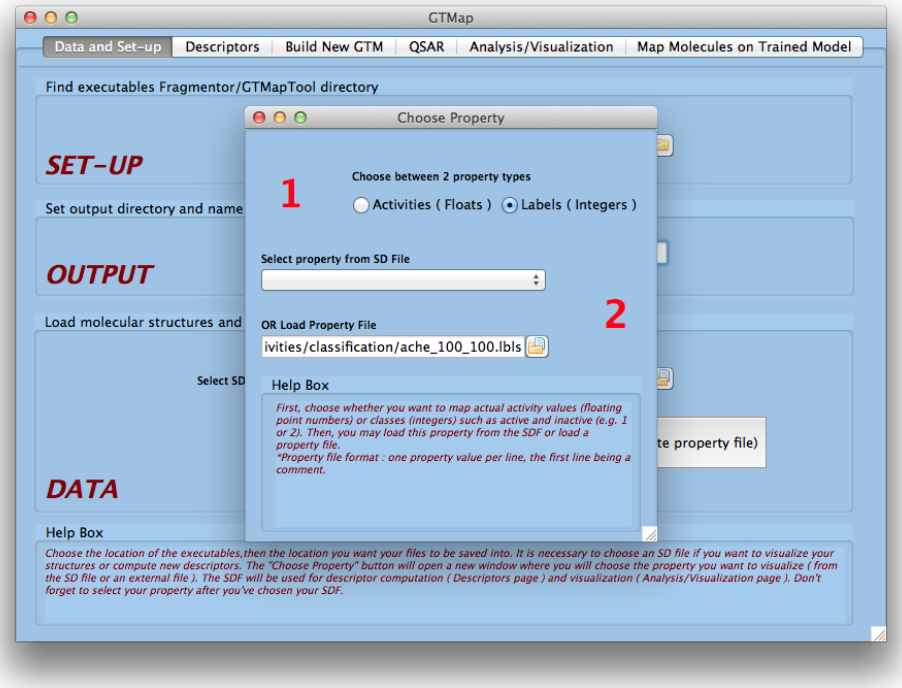


Figure 13.1: *Data and Set-Up tab*, where molecular structures as well as the property to model are loaded.

13.3 Descriptors

The *Descriptors* tab (Figure 13.2) is dedicated to the computation of fragment descriptors. The user may choose the SMF descriptor type he desires or re-compute the same set of descriptors as before by loading a header file, which contains the reference of all fragments. Re-computing the same descriptors is necessary to project new compounds onto an existing map; the dimensionality of the input space must be the same for both training and test sets.

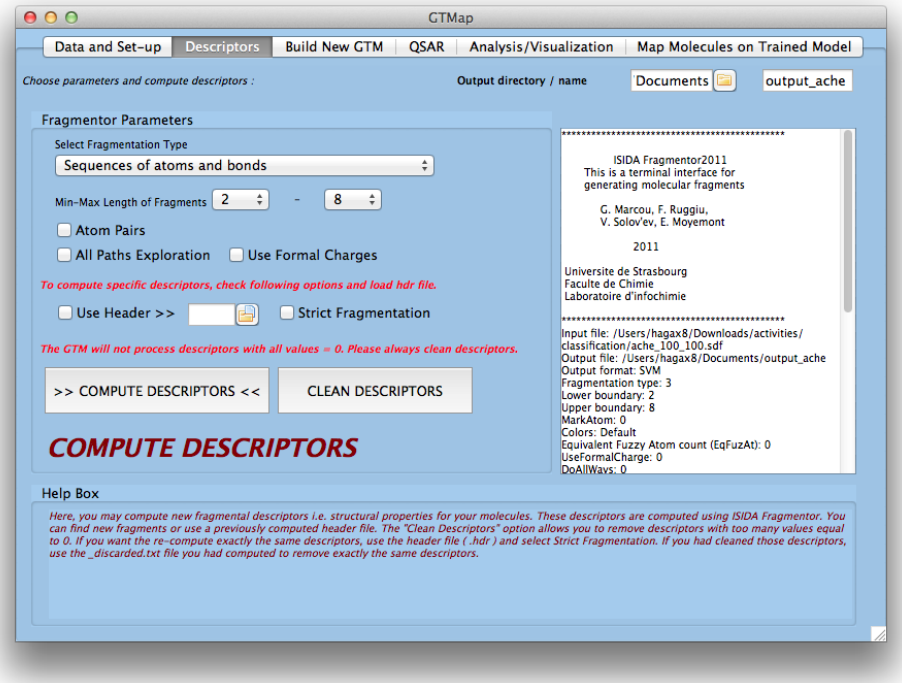


Figure 13.2: *Fragmentor* tab, where descriptors are computed using molecular structures (SDF file).

13.4 Build New GTM

The *Build New GTM* tab (Figure 13.3) is devoted to constructing a GTM model from a descriptor file. The descriptor file computed previously is automatically loaded, but any other external file could be chosen. Four parameters can be tuned: the number of RBF centers M, the number of nodes K, the RBF width factor w and the regularization coefficient λ. In the program, the parameters m and k are chosen so that $m = \sqrt{M}$ and $k = \sqrt{K}$, defining two grids $m \times m$ and $k \times k$; the parameter λ is written l for convenience. In the GUI, the choice of these values was restricted to help the user. The maximum number of iterations can also be set, as well as the log likelihood convergence condition. The user may also choose to compute the coordinates of both data and manifold in 3D PCA space, to see how the manifold fits the data. It is possible to run a batch computation for selecting the model with the best parameters, with or without cross-validation. The log file is displayed in a dedicated panel.

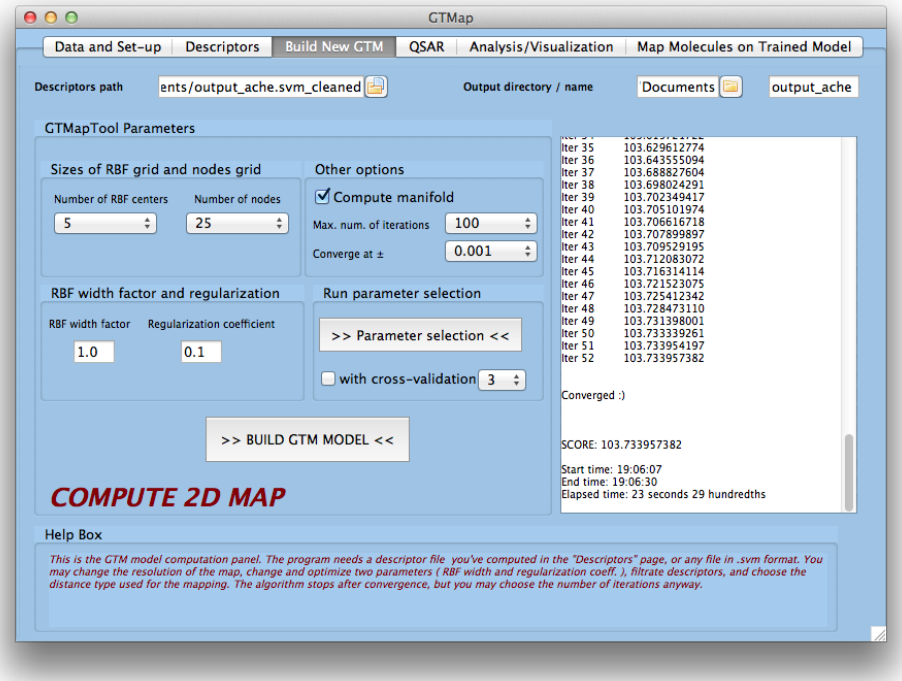


Figure 13.3: *Build New GTM* tab, where the user can run the GTM algorithm using descriptors previously computed or loaded from an external file.

13.5 QSAR

GTM classification and regression models can be constructed in the *QSAR* tab (Figure 13.4). GTM parameters can also be optimized for regression or classification using cross-validation, with any number of folds. The optimized measures for classification are the balanced accuracy and the F-score; for regression models, the performance can be measured either by the determination coefficient or by RMSE. If the goal is not to optimize a model but to predict activities or classes using a GTM regression or classification model, predictions can be generated in the *QSAR* tab.

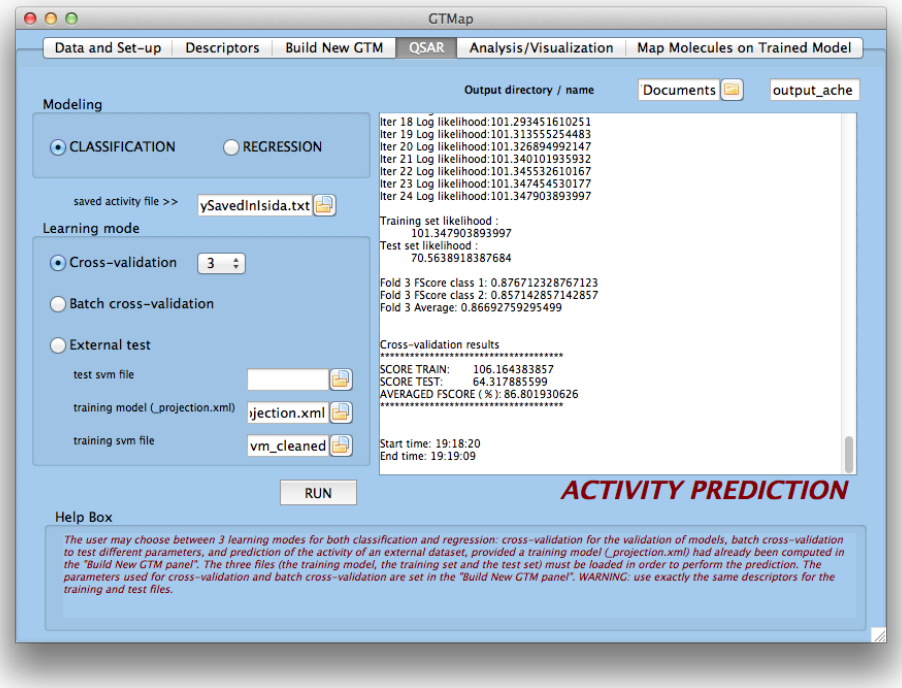


Figure 13.4: *QSAR* tab used for cross-validating GTM-based classification or regression models, and for predicting the activity of new molecules.

13.6 Visualization

The GTM model computed in the *Build New GTM Tab* is visualized with the interactive 2D map visualizer (Figure 13.5). The 2D map visualizer displays the basic GTM map

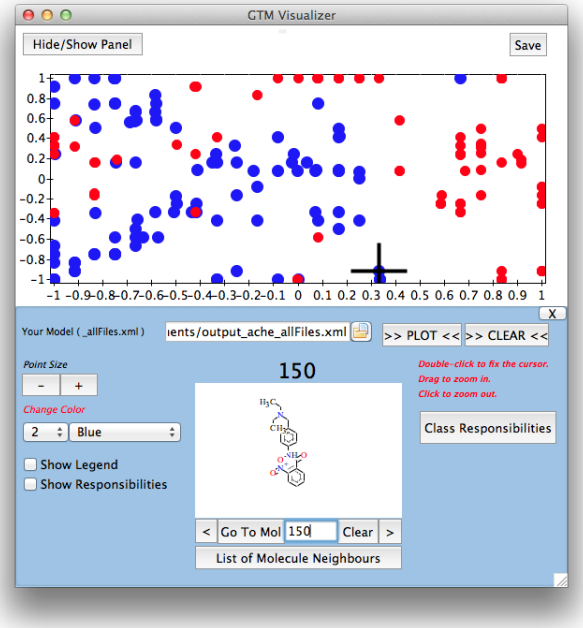


Figure 13.5: GTM 2D visualizer. One point corresponds to one molecule; when the mouse hovers over it, the structure appears in the visualizer pane. This map represents the ache dataset (use case 1), with inhibitors of ache symbolized by red points and decoys by blue points.

where each point corresponds to one molecule. The molecular structure corresponding to each point can be visualized when the mouse hovers over it. It is also possible to zoom in/out on the map, find the closest neighbors of one molecule, and show the distribution of one molecule on the map ("Show Responsibilities" option). Classification maps and activity landscapes can also be created. For such maps, nodes are represented by squares colored by the value of the most probable property (see use cases 1 and 2 for illustrations). The user can visualize the 3D PCA coordinates of both data and manifold in the interactive 3D visualizer (Figure 13.6), where it is possible to click on any data point to visualize the molecular structure in a pop-up window. If the user wants to visualize new molecules on an already built GTM map, he may proceed to the *Map Molecules Onto Trained Model* tab (Figure 13.7) and generate coordinates for the new molecules.

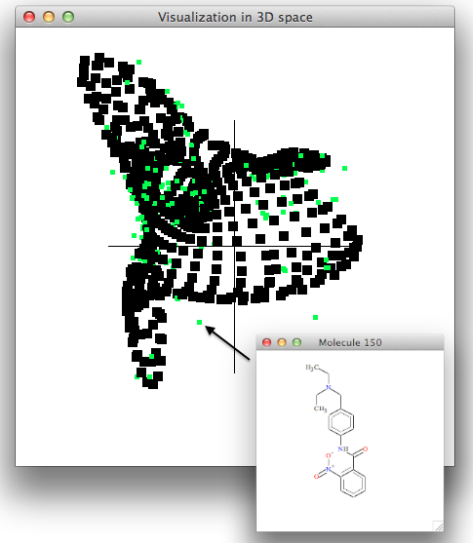


Figure 13.6: Visualization in 3D PCA space of the ache dataset (use case 1). Green points represent molecules and black squares the GTM manifold fitting the dataset.

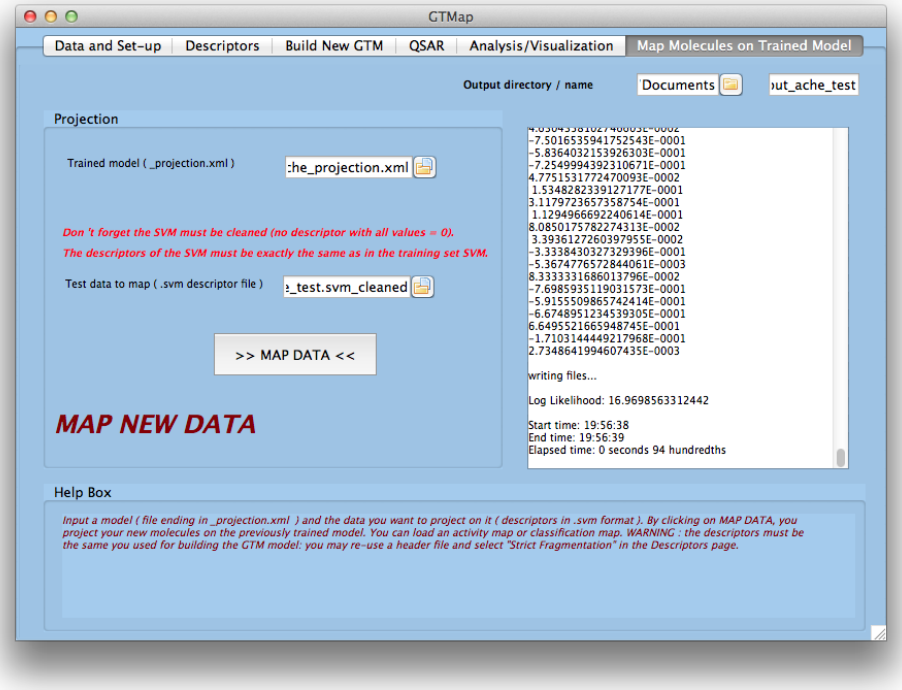


Figure 13.7: In the *Map molecules onto trained model* tab, a model built with a training set can be used to predict the position of new molecules.

13.7 Use Case 1 - Ache Dataset

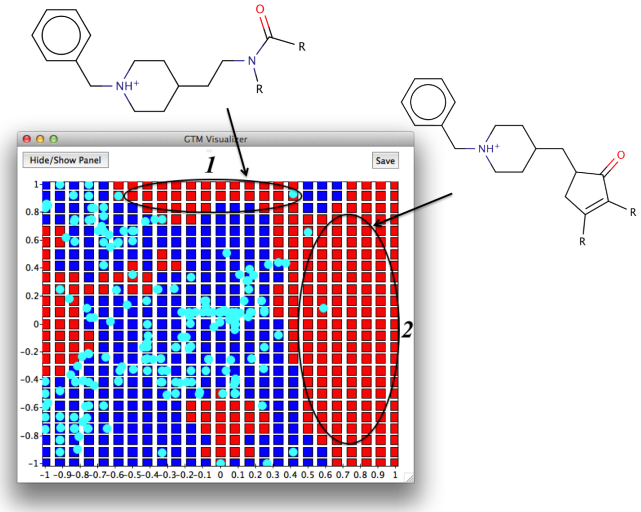


Figure 13.8: Classification map of the ache training set represented by red (active) or dark blue (decoy) squares. The maximum common substructure found in the two best separated active regions (1 and 2) are also shown. The light blue circles represent 200 decoys of an external tet set projected onto this previously trained activity map.

In the implementation section, we used a training set of 100 inhibitors and 100 decoys of acetylcholinesterase (ache), that we took from the DUD (Directory of Useful Decoys) [33]. Inhibitors of ache should typically possess a cationic center to link to the anionic site of the protein and a function to bind its esteratic site [110] which recognizes acetylcholine and therefore is responsible for its hydrolysis. The map shown in Figure 13.5 and built with parameters $[k = 25, m = 5, w = 0.1, l = 1]$ (where k = square root of the number of nodes, m = square root of the number of radial basis functions, w = RBF width factor, l = regularization coefficient) from fragments of atoms and bonds of length 2 to 8, made it possible to see inhibitors forming well-separated clusters and others that where mixed with decoys. Figure 13.8 shows the corresponding classification map, with well-separated clusters of inhibitors characterized by specific common substructures. 3-fold internal cross-validated classification models gave an averaged FScore of 0.87, which shows good predictive performance. An external test set of 200 decoys was projected onto this trained classification map (Figure 13.8); only 5% were mapped onto nodes labeled as active, which means that 95% of these molecules were correctly predicted as decoys by the GTM model.

13.8 Use Case 2 - Gd^{3+} Dataset

The Gd^{3+} dataset containing 308 Gd^{3+} binders will be used to illustrate how the software handles real activity values; for this experiment, the modeled activity is the stability constant β of Gd^{3+}-ligand complexes. After computing atom and bonds fragments of length 2-8, several regression models were computed by 3-fold cross-validation to select the best w and l values for fixed parameters $k = 25$ and $m = 5$, by using the determination coefficient as the objective function (supervised learning). The best R^2 (68%) was obtained with parameters $[k = 25, m = 5, w = 2, l = 100]$. One GTM map was built with the optimized parameters, as well as an activity landscape displayed in Figure 13.9. The activity landscape on Figure 13.9 shows a clear separation between

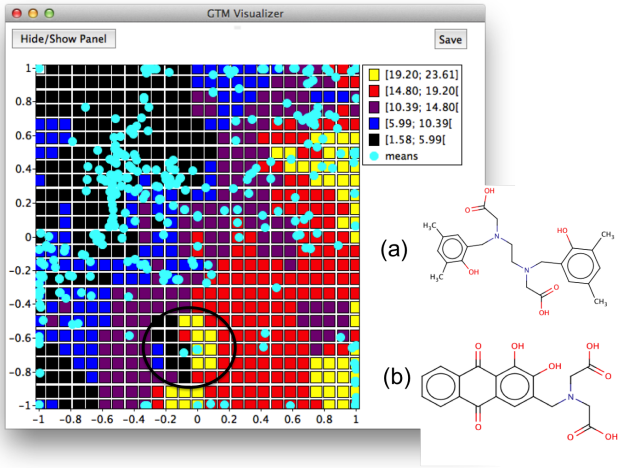


Figure 13.9: Activity landscape built with 308 Gd^{3+} ligands. The squares represent nodes on the GTM grid and their color the value of the stability constant β. The 308 molecules are mapped as light blue circles. An activity cliff is circled, with two molecules on it, mapped next to each other but forming (a) highly stable or (b) less stable complexes.

ligands that form highly stable (high β, yellow-red regions) and unstable (low β, black region) complexes with Gd^{3+}; we can also observe a transition zone for ligands with moderate β. Lanthanides form the most stable complexes with ligands that have a high coordination number (≥ 6). This can be observed on the activity landscape, where the high β zone is populated by ligands with a high coordination number, the lowest β zone by monodentate or bidentate ligands (cf. Figure 13.10 for examples). An activity cliff could also be identified with two ligands forming a stable and an unstable complex with Gd^{3+}, respectively, and mapped next to each other. The presence of this activity cliff could be due to the fact that we used fragment descriptors. Indeed, both ligands have many structural similarities such as the number of aromatic rings, two hydroxyls and two carboxylic acid groups. However, the ligand in the high β zone has an experimental

stability constant $\beta = 20.27$, whereas the ligand in low β zone (alizarin complexone) has an experimental stability constant $\beta = 4.54$. 4 molecules mapped in the low or high β zones are represented on Figure 13.10 with their individual distribution on the map obtained with the software. Ligands (c) and (d) in the high β zone can form more stable chelated complexes and are more localized on the map than molecules (a) and (b) in the low β zone, which form less stable complexes. This is explained by the fact that (a) and (b) have a much less complex structure, and are characterised by fragments which are common to a lot of molecules, therefore their position on the map is more uncertain than that of more complex ligands.

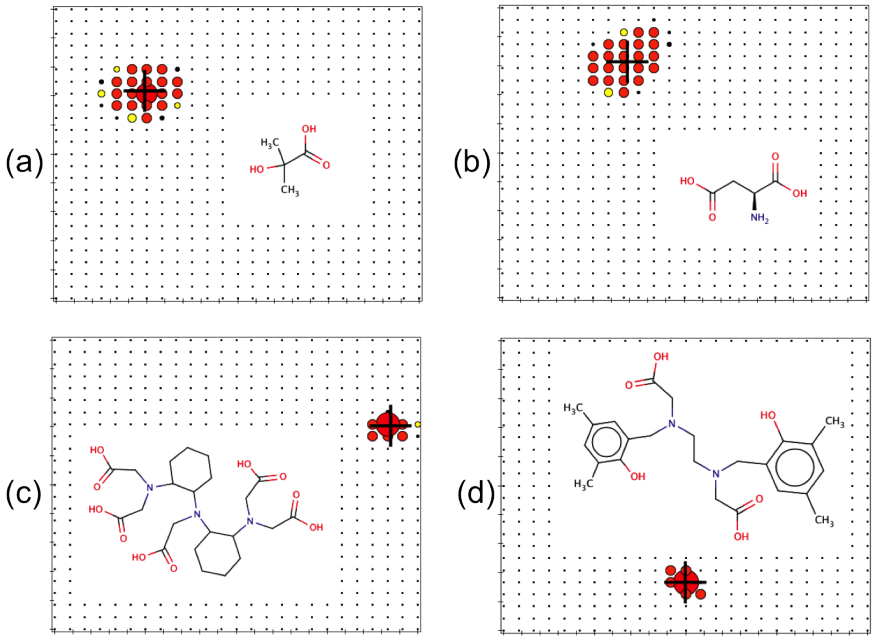


Figure 13.10: ISIDA/GTM representation of the distribution of four individual ligands of Gd^{3+} on an optimized GTM. (a) and (b) form instable complexes, (c) and (d) stable complexes. The size and colour of circles represent the responsibility of each grid node for the represented molecule, and the cross the mean position of the molecule on the conventional GTM.

13.9 Conclusion

ISIDA/GTM is a multi-purpose, visualization-based modelling tool for chemical data. It can be used for QSAR, visualizing clusters, designing class maps and activity landscapes. With the ache dataset, we showed how to visualize regions dominated by different classes, provide an estimation of the predictive performance of the map, observe which substructure characterized a given cluster, and project new molecules on a trained map. With the Gd^{3+} dataset, we demonstrated how to find the best map using cross-validation, study activity landscapes, identify activity cliffs, and see how well the position of a molecule is characterized on a map. We hope that this software will be of interest for chemists, by providing an overview of a chemical dataset and a better understanding of its structure.

Chapter 14

Application of GTM Initial Space Classification to DUD

Chapter 15

Application of GTM Latent Space Classification to BDDCS

Chapter 16

Application of GTM Regression to QSAR

Chapter 17

Application of iGTM to 37 Chemical Libraries

Chapter 18

Conclusion

In this thesis, we explored and brought to light diverse methods for visualizing and analyzing a chemical space, and introduced the new multispace dimensionality reduction technique Stargate GTM, which links a property space to a descriptor space through one unique model.

GTM-based regression and classification techniques exhibit performances close to that of state-of-the-art methods such as Random Forests or SVM. They are neither the fastest nor the most efficient methods but they provide a visual support that many do not offer. The ease of interpretation of visual models compensates for the information loss due to dimensionality reduction. We also introduced several ways of defining the applicability domain of GTM-based models, for both classification and regression tasks, providing a visualization of their boundaries. We used iGTM (incremental GTM) to study large chemical libraries, and introduced various methodologies based on the distribution of libraries on the map to compare and analyze them. Our methods offer a fast and comprehensive way to handle Big Data.

With our Stargate GTM method (S-GTM), we can construct maps trained on both activity and descriptor spaces; we showed on an S-GTM map how a chemical structure usually maps to only one activity (QSAR), and how, on the other hand, an activity maps to several structures (inverse QSAR). Regions on the map corresponding to an activity profile are less extended (lower structural diversity) than regions corresponding to a single activity (higher structural diversity). Both S-GTM and GTM can be used for QSAR or inverse QSAR; however, S-GTM is supervised and more performant. All our developments were implemented into our command-line software GTMapTool, with a graphical user interface GTMap, now part of the ISIDA/GTM software. This software could be a tool for chemists investigating the structure of their data. Other studies involving the GTM approach and chemical data are currently on the way, such as the establishment of a universal map. Moreover, further development of generative inverse

QSAR approaches can be foreseen in the future.

Visualization and chemistry are two topics that have always been closely interwoven. From the time of Democritus to present day, human beings have made assumptions about the structure of matter without being able to observe it; they created models or representations that they could draw and grasp, mental pictures of what their eyes could not see. Models of individual molecular compounds are an aspect of this visualization of matter; models of entire chemical spaces offer a glimpse into the structure of chemical datasets and the relationships between chemical compounds. Some day, we will be able to navigate interactively within a virtual chemical space and to travel between different descriptor space representations; in this thesis, we only scratched the surface of a very large subject. All our models were "static", even when we "traveled" between descriptor and activity spaces with Stargate GTM. An ideal visualization system would update online models of chemical spaces using experimental results in real time, integrating information from various sources, and would allow for user interaction. In other words, a next step for chemical space visualization would be the creation of "living" and adaptive models.

Bibliography

[1] N. Kireeva, I. I. Baskin, H. A. Gaspar, D. Horvath, G. Marcou, and A. Varnek. Generative topographic mapping (GTM): universal tool for data visualization, structure-activity modeling and dataset comparison. *Molecular Informatics*, 31(3-4):301–312, 2012.

[2] H. A. Gaspar, G. Marcou, D. Horvath, A. Arault, S. Lozano, P. Vayer, and A. Varnek. Generative topographic mapping-based classification models and their applicability domain: application to the biopharmaceutics drug disposition classification system (BDDCS). *Journal of Chemical Information and Modeling*, 53(12):3318–3325, 2013.

[3] H. A. Gaspar, I. I. Baskin, G. Marcou, D. Horvath, and A. Varnek. GTM-based QSAR models and their applicability domains. *Molecular Informatics*, 34(6-7):348–356, 2015.

[4] H. A. Gaspar, I. I. Baskin, G. Marcou, D. Horvath, and A. Varnek. Chemical data visualization and analysis with incremental generative topographic mapping: big data challenge. *Journal of Chemical Information and Modeling*, 55(1):84–94, 2015. PMID: 25423612.

[5] H. A. Gaspar, I. I. Baskin, G. Marcou, D. Horvath, and A. Varnek. Stargate gtm: bridging descriptor and activity spaces. *Submitted to the Journal of Chemical Information and Modeling*, 2015.

[6] V. Chupakhin, G. Marcou, H. A. Gaspar, and A. Varnek. Simple Ligand–Receptor Interaction Descriptor (SILIRID) for alignment-free binding site comparison. *Computational and Structural Biotechnology Journal*, 10(16):33–37, 2014.

[7] P. Sidorov, H. A. Gaspar, G. Marcou, D. Horvath, and A. Varnek. Mappability of drug-like space: towards a polypharmacologically competent map of drug-relevant compounds. *Submitted to the Journal of Chemical Information and Modeling*, 2015.

[8] H. G. Liddell and R. Scott. *A Greek-English Lexicon, Ninth Edition with a Revised Supplement.* Clarendon Press, 1996.

[9] C. M. Bishop and C. K. I. Williams. GTM: A principled alternative to the self-organizing map. In *In Advances in Neural Information Processing Systems*, pages 354–360. Springer-Verlag, 1997.

[10] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: the Generative Topographic Mapping. *Neural Computation*, 10(1):215–234, January 1998.

[11] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. Developments of the generative topographic mapping. *Neurocomputing*, 21(1–3):203–224, November 1998.

[12] I. Olier, A. Vellido, and J. Giraldo. Kernel generative topographic mapping. In *ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium)*, 2010.

[13] J. Pokorny. *Indogermanisches etymologisches Wörterbuch.* Francke, 2002.

[14] C. Clapham and J. Nicholson. *The Concise Oxford Dictionary of Mathematics (4th Edition).* Oxford University Press, 2009. Vector Space.

[15] R. Carbó-Dorca. About the concept of Chemical Space: a concerned reflection on some trends of modern scientific thought within theoretical chemical lore. *Journal of Mathematical Chemistry*, 51(2):413–419, 2013.

[16] K. Arai and H. Okazaki. N-Dimensional binary vector spaces. *Formalized Mathematics*, 21(2):75–81, 2013.

[17] R. P. W. Duin and E. Pękalska. The dissimilarity space: bridging structural and statistical pattern recognition. *Pattern Recognition Letters*, 33(7):826–832, 2012.

[18] P. J. Goodford. A computational procedure for determining energetically favorable binding sites on biologically important macromolecules. *Journal of Medicinal Chemistry*, 28(7):849–857, July 1985.

[19] R. D. Cramer, D. E. Patterson, and J. D. Bunce. Comparative molecular field analysis (CoMFA). 1. Effect of shape on binding of steroids to carrier proteins. *Journal of the American Chemical Society*, 110(18):5959–5967, August 1988.

[20] A. Varnek, D. Fourches, F. Hoonakker, and V. P. Solov'ev. Substructural fragments: an universal language to encode reactions, molecular and supramolecular structures. *Journal of Computer-Aided Molecular Design*, 19(9-10):693–703, September 2005.

[21] Chemical Computing Group Inc. Molecular Operating Environment (MOE), 2011.10, 2011.

[22] G. Cruciani. Molecular fields in quantitative structure–permeation relationships: the VolSurf approach. *Journal of Molecular Structure: THEOCHEM*, 503(1-2):17–30, May 2000.

[23] F. Camastra. Data dimensionality estimation methods: a survey. *Pattern Recognition*, 36(12):2945–2954, December 2003.

[24] K. Fukunaga and D.R. Olsen. An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, C-20(2):176–183, February 1971.

[25] R. E. Bellman. *Dynamic Programming*. Courier Corporation, 1957.

[26] G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, January 1968.

[27] M. Zaki. *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, New York, NY, 2014.

[28] M. Shahlaei. Descriptor selection methods in quantitative structure–activity relationship studies: A Review Study. *Chemical Reviews*, 113(10):8093–8103, October 2013.

[29] H. Liu and H. Motoda. *Feature selection for knowledge discovery and data mining*. Springer US, Boston, MA, 1998.

[30] M. L. Bermingham, R. Pong-Wong, A. Spiliopoulou, C. Hayward, I. Rudan, H. Campbell, A. F. Wright, J. F. Wilson, F. Agakov, P. Navarro, and C. S. Haley. Application of high-dimensional feature selection: evaluation for genomic prediction in man. *Scientific Reports*, 5, May 2015.

[31] D. M. Evans, P. M. Visscher, and N. R. Wray. Harnessing the information contained within genome-wide association studies to improve individual prediction of complex disease risk. *Human Molecular Genetics*, 18(18):3525–3531, September 2009.

[32] C. Kooperberg, M. LeBlanc, and V. Obenchain. Risk prediction using genome-wide association studies. *Genetic Epidemiology*, 34(7):643–652, November 2010.

[33] N. Huang, B. K. Shoichet, and J. J. Irwin. Benchmarking sets for molecular docking. *Journal of Medicinal Chemistry*, 49(23):6789–6801, November 2006.

[34] I. Jolliffe. Principal Component Analysis. In *Encyclopedia of Statistics in Behavioral Science*. John Wiley & Sons, Ltd, 2005.

[35] W.S. Torgerson. *Theory and methods of scaling*. Wiley, 1958.

[36] A. S. Householder and G. Young. Matrix approximation and latent roots. *The American Mathematical Monthly*, 45(3):pp. 165–171, 1938.

[37] D. K. Agrafiotis. Stochastic proximity embedding. *Journal of Computational Chemistry*, 24(10):1215–1221, 2003.

[38] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):pp. 321–377, 1936.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[41] G. McLachlan. *Discriminant analysis and statistical pattern recognition*. Wiley-Interscience, Hoboken, N.J, 2004.

[42] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287 – 314, 1994. Higher Order Statistics.

[43] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[44] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999.

[45] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *Advances in Kernel Methods - Support Vector Learning*, pages 327–352. MIT Press, 1999.

[46] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004.

[47] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5):401–409, 1969.

[48] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.

[49] G. De'ath. Extended dissimilarity: a method of robust estimation of ecological distances from high beta diversity data. *Plant Ecology*, 144(2):191–199, 1999.

[50] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science (New York, N.Y.)*, 290(5500):2319–2323, December 2000.

[51] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[52] J. Oksanen, F. G. Blanchet, R. Kindt, P. Legendre, P. R. Minchin, R. B. O'Hara, G. L. Simpson, P. Solymos, M. H. H. Stevens, and H. Wagner. *vegan: Community Ecology Package*, 2015. R package version 2.3-0.

[53] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science (New York, N.Y.)*, 290(5500):2323–2326, December 2000.

[54] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2002.

[55] L. van der Maaten and G. E. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[56] G. Hinton and S. Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15*, pages 833–840. MIT Press, 2002.

[57] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[58] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[59] E. Dubossarsky and Y. Tyshetskiy. *autoencoder: an implementation of sparse autoencoder for automatic learning of representative features from unlabeled data*, 2014. R package version 1.0.

[60] T. Kohonen. *Self-Organizing Maps*. Springer, January 2001.

[61] R. Wehrens and L.M.C. Buydens. Self- and super-organising maps in r: the kohonen package. *J. Stat. Softw.*, 21(5), 2007.

[62] A. Kabán and M. Girolami. A combined latent class and trait model for the analysis and visualization of discrete data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(8):859–872, 2001.

[63] J. R. Owen, I. T. Nabney, J. L. Medina-Franco, and F. López-Vallejo. Visualization of molecular fingerprints. *Journal of Chemical Information and Modeling*, 51(7):1552–1563, 2011.

[64] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistcal Society*, 39(B):1–38, 1977.

[65] S. K. Ng and G. J. McLachlan. On the choice of the number of blocks with the incremental EM algorithm for the fitting of normal mixtures. *Statistics and Computing*, 13(1):45–55, February 2003.

[66] J. Bertin. *La Graphique et le traitement graphique de l'information.* Nouvelle bibliothèque scientifique. Flammarion, Paris, 1977.

[67] G. Sawitzki. *Bertin plots*, 2014. R package version 0.1-94.

[68] F. W. Hewes. General summary, showing the rank of states, by ratios, 1880, 1880.

[69] A. Inselberg. *Parallel Coordinates.* Springer New York, New York, NY, 2009.

[70] P. Hoffman, G. G. Grinstein, K. A. Marx, I. Grosse, and E. Stanley. DNA visual and analytic data mining. In *IEEE Visualization*, pages 437–442, 1997.

[71] D. F. Andrews. Plots of high-dimensional data. *Biometrics*, 28(1):pp. 125–136, 1972.

[72] J. Myslivec. *andrews: Andrews curves*, 2012. R package version 1.0.

[73] Ronald M Pickett and Georges G Grinstein. Iconographic displays for visualizing multidimensional data. *Proceedings of the 1988 IEEE Conference on Systems, Man, and Cybernetics*, 514:519, 1988.

[74] H. P. Wolf and U. Bielefeld. *aplpack: Another Plot PACKage: stem.leaf, bagplot, faces, spin3R, plotsummary, plothulls, and some slider functions*, 2014. R package version 1.3.0.

[75] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, January 1992.

[76] J. LeBlanc, M. O. Ward, and N. Wittels. Exploring n-dimensional databases. In *Proceedings of the 1st Conference on Visualization '90*, VIS '90, pages 230–237, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[77] D.A. Keim and H.-P. Kriegel. VisDB: database exploration using multidimensional visualization. *IEEE Computer Graphics and Applications*, 14(5):40–49, September 1994.

[78] Y. Miyashita, T. Itozawa, H. Katsumi, and S. Sasaki. Comments on the NIPALS algorithm. *Journal of Chemometrics*, 4(1):97–100, January 1990.

[79] Benchmark data sets, trained maps, sample files. `http://www.ifs.tuwien.ac.at/dm/somtoolbox/datasets.html`. Accessed: 2015-06-15.

[80] M. Snarey, N. K. Terrett, P. Willett, and D. J. Wilton. Comparison of algorithms for dissimilarity-based compound selection. *Journal of Molecular Graphics and Modelling*, 15(6):372–385, December 1997.

[81] G. Matheron. Principles of geostatistics. *Economic Geology*, 58(8):1246–1266, December 1963.

[82] M. A. Oliver and R. Webster. A tutorial guide to geostatistics: computing and modelling variograms and kriging. *CATENA*, 113:56–69, February 2014.

[83] F. Mosteller and J. Tukey. Data analysis, including statistics. In G. Lindzey and E. Aronson, editors, *Revised Handbook of Social Psychology*, volume 2, pages 80–203. Addison Wesley, 1968.

[84] C. Rücker, G. Rücker, and M. Meringer. y-Randomization and its variants in QSPR/QSAR. *Journal of chemical information and modeling*, 47(6):2345–2357, September 2007.

[85] B. Efron. Bootstrap methods: another look at the jackknife. *Ann. Statistics*, 7:1–26, 1979.

[86] D. Baumann and K. Baumann. Reliable estimation of prediction errors for QSAR models under model uncertainty using double cross-validation. *Journal of Cheminformatics*, 6(1):47, 2014.

[87] R. Veerasamy, H. Rajak, A. Jain, S. Sivadasan, C. P. Varghese, and R. K. Agrawal. Validation of qsar models - strategies and importance. *International Journal of Drug Design & Discovery*, 3:511–519, 2011.

[88] A. Sen. The impossibility of a paretian liberal. *Journal of Political Economy*, 78(1), 1970.

[89] D. Brümmerhoff. *Finanzwissenschaft*. Oldenbourg, München [u.a.], 7., völlig überarb. aufl. edition, 1996.

[90] J. Neter, W. Wasserman, and M. H. Kutner. *Applied linear statistical models*. Irwin Press, Boston, 1990.

[91] V. Consonni, D. Ballabio, and R. Todeschini. Comments on the Definition of the Q^2 Parameter for QSAR Validation. *J. Chem. Inf. Model.*, 49(7):1669–1678, June 2009.

[92] A. Tropsha. Best Practices for QSAR Model Development, Validation, and Exploitation. *Mol. Inf.*, 29(6-7):476–488, July 2010.

[93] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M. Buhmann. The balanced accuracy and its posterior distribution. In *ICPR*, pages 3121–3124. IEEE Computer Society, 2010.

[94] R. J. Bullen, D. Cornford, and I. T. Nabney. Outlier detection in scatterometer data: neural network approaches. *Neural Networks*, 16(3–4):419 – 426, 2003. Neural Network Analysis of Complex Scientific Data: Astronomy and Geosciences.

[95] A. Sedykh, D. Fourches, J. Duan, O. Hucke, M. Garneau, H. Zhu, P. Bonneau, and A. Tropsha. Human intestinal transporter database: QSAR modeling and virtual profiling of drug uptake, efflux and interactions. *Pharmaceutical Research*, 30(4):996–1007, dec 2012.

[96] C. E Shannon and W. Weaver. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423,623–656, July, October 1948.

[97] Anna Gaulton, Louisa J. Bellis, A. Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, and John P. Overington. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, page gkr777, September 2011.

[98] J. B. Brown, Y. Okuno, G. Marcou, A. Varnek, and D. Horvath. Computational chemogenomics: is it more than inductive transfer? *Journal of Computer-Aided Molecular Design*, 28(6):597–618, April 2014.

[99] G. Marcou, F. Ruggiu, V. Solov'ev, and E. Moyemont. ISIDA Fragmentor, 2013.

[100] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[101] J. Huuskonen. Estimation of aqueous solubility for a diverse set of organic compounds based on molecular topology. *Journal of Chemical Information and Computer Sciences*, 40(3):773–777, 2000.

[102] N. R. McElroy and P. C. Jurs. Prediction of aqueous solubility of heteroatom-containing organic compounds from molecular structure. *Journal of Chemical Information and Computer Sciences*, 41(5):1237–1247, September 2001.

[103] D. Yaffe, Y. Cohen, G. Espinosa, A. Arenas, and F. Giralt. A fuzzy ARTMAP based on Quantitative StructureProperty Relationships (QSPRs) for predicting aqueous solubility of organic compounds. *Journal of Chemical Information and Computer Sciences*, 41(5):1177–1207, September 2001.

[104] L. D. Pettit and K. J. Powell. IUPAC stability constants database (SC-database), 2009.

[105] J. J. Sutherland, L. A. O'Brien, and D. F. Weaver. A comparison of methods for modeling quantitative structure-activity relationships. *Journal of medicinal chemistry*, 47(22):5541–5554, October 2004.

[106] J. V. de Julian-Ortiz. Virtual Darwinian drug design: QSAR inverse problem, virtual combinatorial chemistry, and computational screening. *Combinatorial Chemistry & High Throughput Screening*, 4(3):295–310, May 2001.

[107] D. P. Visco Jr., R. S. Pophale, M. D. Rintoul, and J.-L. Faulon. Developing a methodology for an inverse quantitative structure-activity relationship using the signature molecular descriptor. *Journal of Molecular Graphics and Modelling*, 20(6):429–438, June 2002.

[108] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[109] M. van Canneyt, M. Gärtner, S. Heinig, F. M. de Carvalho, I. Ouedraogo, and J. Braun. *Lazarus Klassenbibliothek und IDE*. Computer & Literatur Verlag GmbH, Böblingen, Germany, 2011.

[110] H. Dvir, I. Silman, M. Harel, T. L. Rosenberry, and J. L. Sussman. Acetylcholinesterase: from 3D structure to function. *Chemico-biological interactions*, 187(1-3):10–22, September 2010.

List of Figures

1 Résumé: concept de la méthode GTM 20
2 Résumé: paysage d'activité 21
3 Résumé: classification GTM 21
4 Résumé: Stargate GTM 24

2.1 Space: etymology 31
2.2 Space: mathematics 32
2.3 Compounds structured in a tree 32
2.4 Chemical universe concept 32
2.5 Descriptor space 33

3.1 Hypersphere v.s. hypercube volume 38
3.2 MDS with different dissimilarity measures 41
3.3 Canonical Correlation Analysis 41
3.4 Linear Discriminant Analysis 42
3.5 Independent Component Analysis 43
3.6 Exploratory Factor Analysis 44
3.7 PCA and kernel PCA 45
3.8 Sammon maps with different dissimilarities 46
3.9 Isomap with different dissimilarities 47
3.10 Locally Linear Embedding 48
3.11 Laplacian Eigenmaps 49
3.12 t-SNE 50
3.13 Autoencoder Diagram 51
3.14 Autoencoder 51
3.15 Self-Organizing Map 52
3.16 GTM concept 57
3.17 GTM and kernel GTM 58
3.18 Incremental GTM 59
3.19 Scatterplot matrix 61

3.20 Bertin matrix. 62
3.21 Parallel coordinates 63
3.22 Andrews curves 63
3.23 Chernoff Faces. 64

4.1 Intertwined circles: data and manifolds 70
4.2 Intertwined circles: manifolds 70
4.3 Intertwined circles: 2D GTM maps 71
4.4 Initial v.s. final log likelihood 71
4.5 Log likelihood as a function of diversity 72

5.1 Activity landscapes 76
5.2 "Min" descriptors 77
5.3 "Max" descriptors 77
5.4 "Sim" descriptors 77
5.5 Descriptor score in GTM node 78
5.6 Regions of interest (ROI) 79

6.1 GTM training and test sets normalization 81
6.2 Model validation 82
6.3 Cross-validation 83
6.4 GTM optimization workflow 84
6.5 GTM parameters 84
6.6 Likelihood Pareto efficiency plot 85

7.1 Classification maps 90

9.1 Outliers 95
9.2 Class prevalence factor applicability domain 98
9.3 Regression applicability domains 100

11.1 S-GTM: outline 108
11.2 S-GTM: training stage 109
11.3 S-GTM: test stage 110
11.4 S-GTM with 1D or 8D activity space 114
11.5 S-GTM QSAR and inverse QSAR 115
11.6 S-GTM QSAR and inverse QSAR (details) 116
11.7 S-GTM performances: 8 affinities 117
11.8 S-GTM performances: MOE descriptors prediction 120
11.9 S-GTM weights: map shape 121

11.10 S-GTM weights: likelihood . . . 121
11.11 Property correlation: Lu . . . 122
11.12 Property correlation: Thrombin . . . 123
11.13 Property correlation: LogS . . . 123
11.14 Property correlation: ChEMBL . . . 124
11.15 Property correlation: individual predictions for Lu S-GTM . . . 124

12.1 Serotonin 1a training and test distribution . . . 128
12.2 Serotonin 1a landscape and regions of interest . . . 129
12.3 Retrieved compounds on 2D GTM (1) . . . 130
12.4 Retrieved compounds on 2D GTM (2) . . . 130
12.5 Retrieved compounds on 2D GTM (3) . . . 131
12.6 Retrieved compounds on 2D GTM (4) . . . 131
12.7 Inverse QSAR with latent prototype approach . . . 132
12.8 Inverse QSAR with GTM manifold prototype . . . 134
12.9 Inverse QSAR with GTM landscape prototype . . . 135
12.10 An activity distribution on the S-GTM . . . 136
12.11 Number of nodes as a function of their activity . . . 137
12.12 Inverse QSAR with GTM weighted landscape prototype . . . 139
12.13 Screening using the conditional log likelihood . . . 140

13.1 Software: data and set-up . . . 145
13.2 Software: descriptors . . . 146
13.3 Software: build new GTM . . . 147
13.4 Software: QSAR . . . 148
13.5 Software: GTM 2D . . . 149
13.6 Software: 3D PCA space . . . 150
13.7 Software: map new data . . . 150
13.8 Software: ache classification . . . 151
13.9 Software: Gd^{3+} activity landscape . . . 152
13.10 Software: molecule distribution . . . 153

List of Tables

3.1 Some linear dimensionality reduction techniques. 40

3.2 Some non-linear dimensionality reduction techniques. 44

3.3 Visual separation of classes on the maps shown in this chapter, measured by balanced accuracy BAC_{FIT}. 54

3.4 Different types of visual structures. 60

4.1 9 different subsets used to initialize the iGTM model: 6 random subsets (r1, r2, r3, r4, r5, r6) and three diverse subsets selected with a diversity-based selection method (d1, d2, d3). 69

4.2 Initial and final log likelihood, when using different initialization subsets with different sizes and proportions from each class (C1/C2); the mean μ and standard deviation σ of the log likelihood for the different subsets is also indicated. 72

11.1 The Affinity dataset contains 1325 molecules with affinities for 8 targets. The target names, ChEMBL IDs and short IDs used in this study are given in the table. Only some compounds had experimentally measured pKi values of affinity for a given target. The number of experimental (exp) and predicted (pred) affinities is indicated. 112

11.2 Datasets used to build conventional and Stargate GTM models. The ChEMBL, LogS, Lu, and Thrombin datasets were used for methodological tests, to build models predicting MOE properties. 118

11.3 Selected parameters [λ, w] for 3-fold cross-validated Stargate GTM regression models, with the corresponding percentage of properties with a determination coefficient R^2 greater than 0.5, the average determination coefficient R^2 and root mean square error $RMSE$ for all properties, and the associated standard deviations. 119

12.1 R^2 and Pearson correlation coefficients for the 3 inverse QSAR method, estimated by comparing the prototype activity (= activity query) to the experimental activity of 1, 3 or 10 best matching structures (BMSs). . . 135
12.2 Pearson correlation coefficient and determination coefficient R^2 comparing activity queries (affinity values ranging from 3 to 10.3 by steps of 0.05), to the experimental activity of 1, 3 or 10 best matching structures or BMSs using S-GTM-based inverse QSAR methods. 141
12.3 R^2 for 4 inverse QSAR methods, estimated by comparing activity queries (sampled from the landscape) to the experimental activity of 1, 3 or 10 best matching structures or BMSs. 141
12.4 R^2 for 4 inverse QSAR methods, estimated by comparing the queries (affinity values ranging from 3 to 10.3 by steps of 0.05) to the experimental activity of 1, 3 or 10 best matching structures or BMSs. 141

Glossary

A

ache

Acetylcholinesterase or ache is an enzyme whose main function is to hydrolyze acetylcholine. 91

activity landscape

In chemoinformatics, an activity landscape represents the activity of compounds on a surface; generally, the x-axis and y-axis represent features of molecules, which can be original descriptors or new variables generated by a dimensionality reduction technique. 2D activity landscapes are usually colored by the activity value; 3D landscapes have a supplementary z-axis representing the activity. 75, 93, 98, 101, 111, 128, 152

AD

An applicability domain (AD) in cheminformatics defines the domain in chemical space where a model's prediction is applicable; it depends on the instances used to train the model and on the features defining the descriptor space. 14, 34, 82, 90, 95, 97–99, 132

Andrews curves

A visualization method where instances are represented by smooth curves on a 2-dimensional plot. 63

autoencoder

An autoencoder is a non-linear dimensionality reduction method based on a neural network that learns to reconstruct a dataset. 51

B

Bayes' theorem

Bayes' theorem is used to estimate conditional probabilities: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$ with $P(B) = \sum_i P(B|A_i)P(A_i)$. 57, 89

big data

The concept of big data arises when the data becomes so large that methods to handle it must be tailored and optimized. 67, 103

BMS

Best matching structure, used in our chapter on inverse QSAR to name retrieved molecular structures corresponding to an activity query. 127

C

canonical variates

In canonical correlation analysis, combination of variables from two spaces that achieve the highest correlation. 42

CCA

Canonical correlation analysis (CCA) is a linear dimensionality reduction method maximizing the correlation between projections in two spaces. 41

CFA

Confirmatory factor analysis (CFA) is a factor analysis method used to test a hypothesized latent structure of a dataset. 44

chemical space

A chemical space is a set of molecular compounds with a specific structure; there are many possible chemical spaces. "The" theoretical Chemical Space encompasses all possible molecular compounds. 31

chemical universe

Set of molecules considered for a given problem. 32

chemoinformatics

Chemoinformatics (or cheminformatics) is a multidisciplinary field linking informatics and chemistry. 82, 95, 105

Chernoff faces

A visualization method encoding descriptors by facial features. 64

cumulated responsibilities

Average responsibilities of nodes for a whole set, computed by averaging the responsibilities of instances within the set. 97, 103

CV

Cross-validation (CV) is a statistical procedure used to validate models, consisting in dividing the training set into n folds; the model is trained with n-1 folds, and evaluated using the remaining "test" fold; the procedure is repeated n times, each time changing the test fold. 82, 83

D

DBCS

Dissimilarity-based compound selection, method to select diverse subsets of compounds. 69

descriptor

A value providing a piece of information on a molecule, such as the number of double bonds, the solubility, etc.; the number of descriptors is equal to the number of apparent dimensions in the dataset. In this thesis, we sometimes use the words "feature", "variable", or "dimension" as related concepts. 32, 35, 36, 38, 67, 75, 76, 81, 96, 105, 107, 127, 143, 146

descriptor space

Vector space used for representing and investigating chemical compounds as vectors, based on specific features called descriptors. 33, 107, 110, 112, 143

dimensionality

The "apparent" dimensionality is equal to the number of dimensions in a dataset; in cheminformatics, it can be the number of original molecular descriptors or features generated by a dimensionality reduction approach. 35–37, 67, 146

dimensionality reduction

Dimensionality reduction consists in generating a small number of informative features from a set of original variables. 34, 37–39, 125

dissimilarity space

Vector space used for representing and investigating chemical compounds as vectors, based on the dissimilarity between compounds. 34

E

EFA

Exploratory factor analysis (EFA) is a linear dimensionality reduction method which identifies latent factors. 43, 44

eigenvalue

Value l by which the eigenvector $\mathbf{v}$ of the linear transformation $\mathbf{A}$ is scaled when $\mathbf{A}$ operates on $\mathbf{v}$: $\mathbf{Av} = l\mathbf{v}$. 37, 47, 49, 56, 65

eigenvector

The non-null vector $\mathbf{v}$ is an eigenvector of the linear transformation $\mathbf{A}$ (square matrix) if its direction does not change when the linear transformation operates on it and is only scaled by a value l (its associated eigenvalue): $\mathbf{Av} = l\mathbf{v}$; the concepts of eigenvalue and eigenvector are used in PCA, where $\mathbf{A}$ is the original covariance matrix. 40, 47, 49, 56

EM

Expectation-maximization (EM) algorithms find in an iterative process the parameters of a probabilistic model corresponding to a local maximum of the likelihood (or log likelihood) function. The E-step (expectation) and M-step (maximization) are repeated until convergence of the likelihood function: the E-step estimates posterior probabilities and the M-step estimates parameters using the new posterior probabilities. 68

F

feature selection

Process of selecting relevant features from a set of descriptors. 38, 67

G

GTM

Generative topographic mapping (GTM) is a non-linear dimensionality reduction method, and the probabilistic counterpart of SOM. 39, 53, 55

GTMapTool

Command-line tool developed during this thesis for constructing GTM, kernel GTM, incremental GTM and Stargate GTM models, including various options for building and validating classification and regression models. 58, 59, 143, 163

GUI

A graphical user interface (GUI) is an interface between the user and the machine that uses graphical elements such as windows, buttons, etc. 144

I

ICA

Independent component analysis (ICA) is a linear dimensionality reduction method that maximizes the independence of signal components. 43

IDE

An integrated development environment (IDE) is a set of tools integrated in one single program to help develop softwares. 144

iGTM

Incremental generative topographic mapping (iGTM) is an incremental variant of GTM for handling large datasets. 13, 20, 23, 55, 58, 67, 68

InfoVis

Information visualization, a scientific field dedicated to visual representations of information. 13, 60, 61

intrinsic dimensionality

The intrinsic dimensionality is the minimum number of features needed to describe a dataset given an initial set of descriptors (features). 36, 58

inverse QSAR

Inverse quantitative structure-activity relationship (inverse QSAR) models seek to find relationships between structural and activity information; an inverse QSAR analysis consists in two steps: 1. generate descriptors from activities, 2. generate structures from descriptors; in this thesis, the first step was performed to find existing structures, without generating new chemical compounds. 34, 53, 114, 116, 127

ISIDA

ISIDA stands for "in silico design and data analysis", a project carried out by the Laboratory of Chemoinformatics (Université de Strasbourg, France). 36, 39

ISIDA/GTM

Toolbox developed during this thesis for constructing GTM, kernel GTM, incremental GTM and Stargate GTM models including the command-line tool GTMapTool and the graphical user interface GTMap. The software is part of the ISIDA project carried out by the Laboratory of Chemoinformatics (Université de Strasbourg, France). 143, 163

Isomap

Isomap is a non-linear dimensionality reduction method based on the eigendecomposition of a geodesic distance matrix. 47

K

kernel

The kernel function k can be seen as a measure of similarity; it will be qualified as kernel if it is symmetric ($k(x_i, x_j) = k(x_j, x_i)$), and if the kernel matrix $\mathbf{K}$ with elements $K_{ij} = k(x_i, x_j)$ is positive semi-definite so that: $\sum_i \sum_j K_{ij} c_i c_j \geq 0$ where c_i and c_j are real numbers. The kernel matrix is also called Gram matrix. 33, 41, 45–47, 49, 55, 57, 67, 143

kernel trick

With the kernel trick, it is possible to map the data $\mathbf{t}$ into a space of higher dimensionality (feature space) with a function $\phi(\mathbf{t})$ that is never computed, by reducing the inner product between objects in feature space to a kernel function in input space: $K(\mathbf{t}_i, \mathbf{t}_j) = \phi(\mathbf{t}_i)\dot{\phi}(\mathbf{t}_j)$. 45

KGTM

Kernel GTM (KGTM) is a kernel variant of GTM. 20, 57, 67, 143

KPCA

Kernel PCA (KPCA) is a non-linear dimensionality reduction method using a kernel instead of the original data. 45

L

landscape prototype

Term introduced in our inverse QSAR chapter to define a structure prototype composed of descriptor landscape values defining a "prototype" molecule. 134, 190

Laplacian Eigenmaps

The Laplacian Eigenmaps approach is a non-linear dimensionality reduction method based on the eigendecomposition of the graph Laplacian. 49

latent prototype

Term introduced in our inverse QSAR chapter to define a structure prototype composed of manifold coordinates in the latent space defining a latent "prototype" molecule. 128, 190

latent space

The latent space is a space defined by "hidden" variables; the goal of many dimensionality reduction algorithms is to find a few of these latent or hidden variables (two or three for visualization purposes) from a set of observed variables (descriptors); in this thesis, the dimensionality of the latent space is most of the time equal to two. 50, 76, 89, 108

LDA

Linear discriminant analysis (LDA) is a dimensionality reduction method and a supervised classifier. 42

linear dimensionality reduction

Process of reducing the dimensionality of a dataset through a linear mapping. 39

LLE

Locally linear embedding (LLE) is a non-linear dimensionality reduction method that assumes that data points in the initial space and their neighbors are close to a locally linear patch of manifold. 48

log likelihood

Objective function $\mathcal{L}$ used in maximum likelihood estimation to choose the parameters θ optimizing the probability that the model will generate the data: $\mathcal{L} = \sum_i \ln p(\mathbf{t}_i|\theta)$, where $p(\mathbf{t}|\theta)$ is a probability distribution over instances $\mathbf{t}_i$; in this thesis, we always used the normalized log likelihood by dividing $\mathcal{L}$ by the number N of instances. 53, 56, 60, 73, 85, 97, 139, 147

LTM

A latent trait model (LTM) uses categorical data to find continuous latent variables; a latent trait analysis can be used to reduce the number of dimensions of a binary or categorical dataset. 55

M

manifold

A manifold is a topological space that is locally Euclidean, *i.e.*, which resembles the Euclidean space in the neighborhood of each of its points; a line is a 1-dimensional manifold, a surface is a 2-dimensional manifold. Many dimensionality reduction methods use the concept of manifold to model a multidimensional dataset; the manifold is then often a 2-dimensional surface embedded in the multidimensional space. The shape of the manifold should approximate the shape of the dataset. 37, 39, 45, 48, 50, 55, 68, 85, 89, 93, 108, 133, 147

manifold prototype

Term introduced in our inverse QSAR chapter to define a structure prototype composed of manifold coordinates in the D-dimensional descriptor space defining a "prototype" molecule. 133, 190

MDS

Multidimensional scaling (MDS) is a type of dimensionality reduction, comprising metric and non-metric MDS and representing the similarity of molecules in a reduced space. Metric MDS respects the input space distances and non-metric MDS the distance rankings. The classical metric MDS is a linear dimensionality reduction method. 40, 46, 47, 50

N

NNMF

Non-negative matrix factorization (NNMF) is a linear dimensionality reduction approach for non-negative input matrices. 44

non-linear dimensionality reduction

Process of reducing the dimensionality of a dataset, usually by trying to discover a non-linear manifold embedded in the data space. 44, 55

NPD

Normal probability distribution. 56, 57

O

outlier

In statistics, a data point which significantly differs from the other data points; it could be seen as lying in the tail of a probability distribution; if it is a molecule, it may differ in terms of structure, activity, or both structure and activity. 39, 95

overfitting

Overfitting is a phenomenon arising when a model fits the training data too well, in such a way that its performance on new data decreases as the performance on the training data increases. 38, 85

P

parallel coordinates

A visualization method where instances are represented by lines on a 2-dimensional plot; the y-axis gives the value of the descriptor that can be found on the x-axis. 60, 62–64

PCA

Principal component analysis (PCA) is a linear dimensionality reduction method that produces uncorrelated features. 40, 56, 68, 147

PDF

A probability density function (PDF) is a function f of a continuous random variable X such that: $P(a \leqslant X \leqslant b) = \int_a^b f(x)dx$. 89, 94, 109, 110

Q

QSAR

Quantitative structure-activity relationship (QSAR) models aim at finding relationships between structural and activity information. 57, 82, 85, 93, 114, 115, 144, 148

R

RBF

A radial basis function (RBF) can be any type of real-valued function depending only on the distance from a center. 46, 53, 56, 58, 68, 83, 112, 147

RF

Random forests, a machine learning technique based on multiple decision trees. 117

RLE

The relative landscape elevation (RLE) was introduced in this thesis to study activity landscapes; it measures the percentage of the map where a specific property has higher values than a reference. 104

ROI

In this thesis, a region of interest (ROI) is a delimited area in the 2D latent space matching several criteria; we usually obtain ROIs by superimposing activity landscapes, *e.g.*, logS and chirality landscapes. 75, 79, 133

S

S-GTM

Stargate GTM, a variant of GTM for several descriptor spaces. 14, 15, 34, 94, 107, 111, 135, 143

Sammon mapping

Sammon mapping is a non-linear dimensionality reduction method based on preserving the input space distances. 46

SMF

Substructural molecular fragment, a type of molecular descriptor. 36

SNE

Stochastic neighbor embedding (SNE) is a non-linear dimensionality reduction method, based on minimizing the divergence between joint probabilities in the input and latent space. 50

SOM

The self-organizing map (SOM) or Kohonen map is a non-linear dimensionality reduction method based on a neural network. 44, 52, 53

structure prototype

Term introduced in our inverse QSAR chapter to define a set of estimated structural descriptors or "prototype" molecule; it includes D-dimensional prototypes

(see manifold prototype or landscape prototype) defined in the D-dimensional descriptor space and 2D prototypes (see latent prototype) defined in the latent space. 127

T

t-SNE

t-distributed stochastic neighbor embedding (t-SNE) is a non-linear dimensionality reduction method derived from SNE, based on minimizing the divergence between joint probabilities in the input and latent space, where probabilities in the input space are computed using a Student t-distribution. 50

Symbols (GTM and variants)

$\mathbf{A}$	Activity profile query (inverse QSAR)
$\mathbf{A}'$	Retrieved activity profile (inverse QSAR)
$\mathbf{a}$	Vector of activities
$\bar{\mathbf{a}}$	K-dimensional activity landscape vector
$\hat{\mathbf{a}}$	Vector of estimated activities
β	Inverse variance of GTM normal probability distributions
c_i	Depending on the context, ith class or ith library
D	Number of dimensions, dimensionality
E_q^{class}	Class entropy of an instance (molecule) $\mathbf{t}_q$ based on class probabilities
$E_{\text{norm}}(c_i)$	Normalized entropy of a set of instances c_i based on cumulated responsibilities
$\mathbf{\Phi}$	$M \times K$ matrix of M radial basis functions
$\mathbf{G}$	$K \times K$ matrix where $G_{kk} = \sum_n R_{kn}$
$\mathbf{I}$	Identity matrix
K	Number of grid nodes
L_{test}	Test set likelihood evaluated in cross-validation
L_{train}	Training set likelihood evaluated in cross-validation
$\mathcal{L}(\mathbf{W}, \beta)$	Log likelihood
$\mathcal{L}_{norm}(\mathbf{W}, \beta)$	Normalized log likelihood
$\mathcal{L}_n(\mathbf{W}, \beta)$	Log likelihood of the nth compound
$\mathcal{L}_{qa}$	Conditional log likelihood used as a scoring function for the qth compound and ath activity query
λ	Regularization coefficient
M	Number of RBF centers
$\boldsymbol{\mu}_m$	2D coordinates of the mth RBF center
N	Number of molecules
N'	Number of molecules in a data block (iGTM)
N_c	Number of classes

Symbols (GTM and variants)

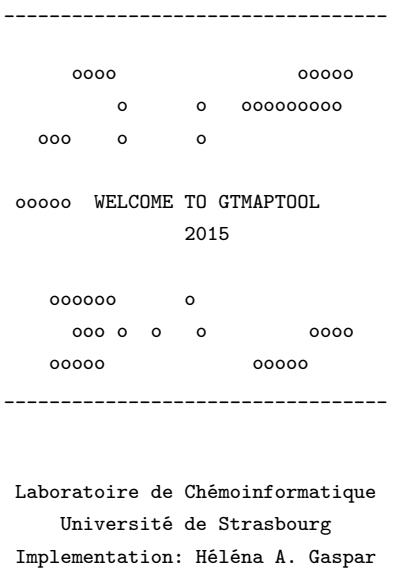
N_{c_i}	Number of compounds belonging class c_i
$\mathbf{O}$	Original $N \times D$ data matrix in D-dimensional space before normalization or standardization
$\mathbf{o}^d$	N-dimensional vector containing the dth column of the $\mathbf{O}$ matrix (all instances for one descriptor)
Q^2	Cross-validated determination coefficient
$\mathbf{R}$	$K \times N$ responsibilities matrix
$\mathbf{R}'$	$K \times N'$ responsibilities matrix of a data block (iGTM)
$\bar{\mathbf{R}}$	$N_C \times K$ matrix of cumulated responsibilities of K nodes for N_C sets
R^2	Determination coefficient
$\mathbf{S}$	Structure query (inverse QSAR)
$\mathbf{S}'$	Retrieved structure (inverse QSAR)
$S(c_i, c_j)$	Similarity (or dissimilarity, depending on the chosen measure) between two sets c_i and c_j
σ^2	RBF parameter
σ_k^2	Activity variance in node $\mathbf{x}_k$
$\mathbf{T}$	$N \times D$ data matrix in D-dimensional space after normalization or standardization
$\mathbf{T}'$	$N' \times D$ matrix representing a data block (iGTM)
$\mathbf{t}_n$	D-dimensional vector representing a data instance
τ	Average squared Euclidean distance between RBF centers
$\mathbf{U}$	$D \times L$ matrix of L first eigenvectors of the data ($\mathbf{T}$) covariance matrix
$\mathbf{W}$	$D \times M$ parameter matrix
w	RBF width factor
w^{Space1}	Weight attributed to Space 1 (S-GTM)
w^{Space2}	Weight attributed to Space 2 (S-GTM)
$\mathbf{X}$	$K \times 2$ matrix of manifold coordinates in 2-dimensional latent space
$\mathbf{X}(\mathbf{T})$	$N \times 2$ matrix of mean positions of data instances in the 2D latent space
$\mathbf{x}_k$	2D coordinates of the kth node on the manifold
$\mathbf{x}(\mathbf{t}_n)$	mean position of the nth data instance in the latent space
$\mathbf{Y}$	$K \times D$ matrix of manifold coordinates in D-dimensional space
$\mathbf{y}_k$	D-dimensional coordinates of the kth node on the manifold
$y(\mathbf{x}; \mathbf{W})$	Function performing a non-linear mapping of points $\mathbf{x}$ from the 2D latent space into the D-dimensional data space

Appendix A

Visualization of the SILIRID Space

Appendix B

GTMapTool 2015 Software Options



```
//////////////
Basic Parameters
//////////////

-h get help (no argument)
-i svm input file (-i myfile.svm)
-o output file (-o out)
-w width factor for radial basis functions (-w 1.5)
-m square root of the number of RBF centers (-m 5 for 25 centers)
-k square root of the number of grid points (-k 25 for 625 points)
-l regularization coefficient, usually between 0.0001 and 1000 (-l 10.0)
-r random initialization of W matrix (GTM parameters) (no argument)
-f apply PCA before GTM (-f 20 for selecting 20 variables)
-t input .dat instead of .svm with (tab-separated values) (no argument)
-p clean input file by selecting descriptors with a minimum % of non-null values
(-p 0.01 for at least 1% of non-null values)
-c number of iterations (-c 100)
-b find best values for -l, -w, -k and -m (no argument)
-v perform crossvalidation (-v 3 for 3 folds)
-j perform projection from model _projection.xml (-j -y model_projection.xml)
-y input model for projection (-j -y model_projection.xml)
-a compute manifold and data coordinates in PCA 3D space (no argument)
-n clean file (no argument)

//////////////
Verbose Parameters
//////////////

--seed
different seed for cross-validation
(--seed=3)

--batchW
define set of values for k to use with -b or -b -v; syntax: begin:step:end
(--batchW=0.5:0.5:2)

--batchL
define set of values for log10(l) to use with -b or -b -v; syntax: begin:step:end
(--batchL=-2:1:2)

--batchM
define set of values for m to use with -b or -b -v; syntax: begin:step:end
(--batchM=4:1:6)

--batchK
define set of values for k to use with -b or -b -v; syntax: begin:step:end
(--batchK=10:5:20)

--limit
convergence criterion
(--limit=0.001)
```

```
--classes
labels (one value per line, first line is a comment)
(--classes=labelFilePath.txt)

--activities
activities (one value per line, first line is a comment)
(--activities=activityFilePath.txt)

--test
external test set (--test=externalTestSetPath.svm)

--magnificationFactors
compute magnification factors for classical GTM
(--magnificationFactors)

--noNormalization
do not normalize data
(--noNormalization)

--scaling
use file with min and max values for normalizing descriptors
(-n --scaling=myoutput_scaling.txt)

-n -y
use file for discarding descriptors
(-n -y myoutput_discarded.txt)

--missing
computes missing values with attribute value NAN in svm file
(format: 1:NAN) (no argument)

--NIPALS
use NIPALS algorithm for PCA: speeds up things for higher dimensions
(no argument)

--kernel
kernel algorithm, choose tanimoto, gaussian, linear, etc.
(--kernel=tanimoto or --kernel=gauss or --kernel=linear, etc.)

--kernelFile
input kernel for kernel GTM
(--kernelFile=kernelTrain.txt)

--kernelCentering
center kernel
(--kernelCentering)

--cste
constant for linear and polynomial kernels, default is 0
(--cste=0)
```

```
--degree
degree for polynomial kernels, default is 2
(--degree=2)

--slope
slope for polynomial kernels, default is 1
(--slope=1)

--nDim
for kernel GTM, feature space dimension
(--kernel=tanimoto --nDim=50)

--gamma
for kernel GTM with gaussian kernel exp(-gamma*x*x), gamma coefficient
(--kernel=gauss --gamma=0.5)

--computeKernels
compute 3 kernels: train*train, test*test, train*test
(--computeKernels --train=x --test=y --kernel=gauss)

--incremental
incremental algorithm... don't forget to set the --blocks and -c options
(--incremental)

--blocks
number of blocks (packs of molecules) for the incremental algorithm
(--blocs=11)

--initialize
number of molecules used in the initialization process (first ones)
(--initialize=2000)

--inputFiles
for the incremental algorithms, when the dataset is divided into several files
(--inputFiles=file1.svm:file2.svm:file3.svm)

--combinedGTM
GTM with 2 manifolds for descriptors and properties (properties begin at ID x)
(--combinedGTM=50)

--COVAR
use entire covariance matrix instead of initial set for incremental GTM initialization
(--COVAR)

--PCAINCR
incremental PCA
(--PCAINCR)
```

```
//////////
Examples
//////////

- Simple GTM:
$ GTMapTool -i mySvm.svm -o output -k 15 -m 5 -w 0.5 -l 10

- Batch GTM:
$ GTMapTool -i mySvm.svm -o output -k 15 -m 5 -b --batchW=0.5:0.5:2 --batchL=-2:1:2

- Projection:
$ GTMapTool -i mySvm.svm -o output -j -y myModel_projection.xml

- 3-fold cross-validation:
$ GTMapTool -i mySvm.svm -k 15 -m 5 -o output -v 3

- Batch 5-fold cross-validation:
$ GTMapTool -i mySvm.svm -o output -b -v 5

- GTM classification with cross-validation:
$ GTMapTool -i myWholeDataset.svm -o output --classes=labelFile.txt -v 3 -m 5 -k 15 -w 0.1 -l 1.0

- GTM classification with batch cross-validation (test different parameters l and w):
$ GTMapTool -i myWholeDataset.svm -o output --classes=labelFile.txt -v 3 -b -m 5 -k 15

- Predict labels of external test set with trained model:
$ GTMapTool -i myTrainingDataset.svm -y myTrainedModel_projection.xml -o output \
--classes=classesOfTrainedModel.txt --test=myTestDataset.svm

- GTM regression with cross-validation:
$ GTMapTool -i myWholeDataset.svm -o output --activities=activityFilePath.txt -v 3

- GTM regression with batch cross-validation (test different parameters l and w):
$ GTMapTool -i myWholeDataset.svm -o output --activities=activityFilePath.txt -v 3 -b -m 5 -k 15

- Predict activities of external test set with trained model:
$ GTMapTool -i myTrainingDataset.svm -j -y myTrainedModel_projection.xml -o output \
--activities=activitiesOfTrainedModel.txt --test=myTestDataset.svm

- Clean input file by removing descriptors given in _discarded.txt
and normalizing with _scaling.txt boundaries:
$ GTMapTool -i mySvm.svm -o output -n -y myoutput_discarded.txt --scaling=myoutput_scaling.txt

- Clean input file by removing descriptors with less then 10% non-null values,
without the default attribute normalization:
$ GTMapTool -i mySvm.svm -o output -n -p 0.1 --noNormalization

- Perform PCA and obtain 50 attributes (removes null descriptors beforehand):
$ GTMapTool -i mySvm.svm -o output -n -f 50
```

```
- Make GTM and fill missing values (NAN in .svm file e.g. 3:NAN):
$ GTMapTool -i mySvm.svm -o output --missing

- Incremental learning (< 200 000 molecules for 200 descriptors and 8 GO RAM;
small dataset and dimensions < 1000):
$ GTMapTool -i mySvm.svm -o output -k 15 -m 5 --incremental --blocks=11 -c 10

- Incremental learning (large dataset but < 200000,
high dimensionality, randomize mySvm.svm beforehand!):
$ GTMapTool -i mySvm.svm -o output -k 15 -m 5 \
--incremental --blocks=11 -c 10 --initialize=100 --NIPALS

- Incremental learning with several files,
high dimensionality (randomize files beforehand!):
$ GTMapTool --inputFiles=file1.svm:file2.svm:file3.svm -o output -k 15 -m 5 \
--incremental --blocks=11 --initialize=100 --NIPALS

- Kernel GTM with svm input file:
$ GTMapTool -i mySvm.svm -o output --kernel=tanimoto -c 100 --nDim=20

- Kernel GTM with kernel input file:
$ GTMapTool --kernelFile -i myKernel.txt -o output -c 100 --nDim=20

- Compute kernels for training and test set, train*train, test*test and test*train:
$ GTMapTool --computeKernels -o output --train=train.svm --test=test.svm --kernel=tanimoto

- Kernel projection:
$ GTMapTool -j -y myTrainedModel_projection.xml -o output --kernel=tanimoto \
--test=testKernel.txt --train=trainKernel.txt --testVtrain=testTrainKernel.txt

- Predict activities of test set with trained model and train-test kernels with kernel GTM:
$ GTMapTool -j -y myTrainedModel_projection.xml -o output \
--activities=activitiesOfTrainedModel.txt --kernel=tanimoto --nDim=50 \
--test=testKernel.txt --train=trainKernel.txt --testVtrain=testTrainKernel.txt

- Predict labels of test set with trained model and train-test kernels with kernel GTM:
$ GTMapTool -j -y myTrainedModel_projection.xml -o output \
--classes=classesOfTrainedModel.txt --kernel=tanimoto --nDim=50 \
--test=testKernel.txt --train=trainKernel.txt --testVtrain=testTrainKernel.txt

- Perform combined GTM ("S-GTM") with properties beginning at 10, 20% weight for descriptors:
$ GTMapTool -i mySvm.svm --combinedGTM=10 -o output --weight=0.20

- Perform reverse mapping: obtain all coordinates in space 2 (trained with combined GTM):
$ GTMapTool -i space1_data.svm -o output -j -y space1_projection.xml \
--reverseProjection=space2_reverseProjection.xml

- GTM for binary data: LTM (Latent Trait Model)
$ GTMapTool -i mySvm.svm -o output --BIN --noNormalization
```

```
//////////
Input File Formats
//////////

- svm file:
=> no comment line, one line is an individual, described by descriptors
with IDs and values represented as a couple "ID:value":
ONE LINE = ? 1:2 2:3 4:5
where ? can be any value and is not taken into account in this program

- tab-separated file:
=> no comment line, alternative to svm, with -t option:
ONE LINE = 2 3 0 5

- input kernel file:
=> complete Gram matrix, first line is a comment, values are separated by commas:
ONE LINE = 1,0.1,0.2,0.3

- activity or class file:
=> first line is a comment, each line is the activity of one instance:
ONE LINE = 0.5


//////////
Output Files for Conventional GTM
//////////

- out_matMeans.txt:
coordinates of points (each point = one object) as mean positions in 2D latent space

- out_matModes.txt:
coordinates of points (each point = one object) as mode positions in 2D latent space

- out_matX.txt:
coordinates of nodes (grid points in 2D latent space)

- out_matR.txt:
posterior probabilities or "responsibilities" of nodes (columns) for each object (row)

- out_allFiles.xml:
contains all previous files, to be used in GUI

- out_projection.xml:
trained model for projecting new data into the 2D latent space
```

```
//////////
Output files specific to 2-spaces GTM
//////////

- out_prop_reverseProjection.xml:
trained model used to map data from 2D latent space into N-dimensional property space

- out_desc_reverseProjection.xml:
trained model used to map data from 2D latent space into N-dimensional descriptor space

- out_prop_projection.xml:
trained model used to map data from property space into the 2D latent property space

- out_desc_projection.xml:
trained model used to map data from descriptor space into the 2D latent descriptor space

- out_dataInPropertySpace.svm:
data projection in N-dimensional property space (can be used as property predictions)
```