Algorithmes d'apprentissage automatique pour la conception de composés pharmaceutiques et de vaccins

Thèse

Sébastien Giguère

Doctorat en informatique
Philosophiæ doctor (Ph.D.)

Québec, Canada

Résumé

La découverte de composés pharmaceutiques est actuellement trop longue et trop coûteuse, et le taux d'échec, trop élevé. Les bases de données biochimiques et génomiques ne cessent de grossir et il est maintenant impraticable d'interpréter ces données. Un changement radical est nécessaire ; certaines étapes de ce processus doivent être automatisées.

Les peptides jouent un rôle important dans le système immunitaire et dans la signalisation cellulaire. Leurs propriétés favorables en font des candidats de choix pour initier la conception de nouveaux médicaments et assister la production de nouveaux vaccins. De plus, les techniques de synthèse modernes permettent de rapidement synthétiser ces molécules à faible coût.

Les algorithmes d'apprentissage statistique sont particulièrement bien adaptés pour apprendre de façon automatisée des modèles, possiblement biochimiques, à partir des données existantes. Ces méthodes et les peptides offrent donc une solution de choix aux défis auxquels fait face la recherche pharmaceutique.

Nous proposons un noyau permettant l'apprentissage de modèles statistiques de phénomènes biochimiques impliquant des peptides. Celui-ci permet, entre autres, l'apprentissage d'un modèle universel pouvant raisonnablement quantifier l'énergie de liaison entre toute séquence peptidique et tout site de liaison d'une protéine cristallisée. De plus, il unifie la théorie de plusieurs noyaux existants tout en conservant une faible complexité algorithmique.

Ce noyau s'avère particulièrement adapté pour quantifier l'interaction entre les antigènes et les complexes majeurs d'histocompatibilité. Nous proposons un outil pour prédire les peptides qui survivront au processus de présentation antigénique. Cet outil a gagné une compétition internationale et aura plusieurs applications en immunologie, dont la conception de vaccins.

Ultimement, un peptide doit maximiser l'interaction avec une protéine cible ou maximiser la bioactivité chez l'hôte. Nous formalisons ce problème comme un problème de prédiction de structures. Puis, nous proposons un algorithme exploitant les plus longs chemins dans un graphe pour déterminer les peptides maximisant la bioactivité prédite par un modèle préalablement appris. Nous validons cette nouvelle approche en laboratoire par la découverte de peptides antimicrobiens. Finalement, nous fournissons des garanties de performance de type PAC-Bayes pour deux algorithmes de prédiction de structure dont un est nouveau.

Abstract

The discovery of pharmaceutical compounds is currently too time-consuming, too expensive, and the failure rate is too high. Biochemical and genomic databases continue to grow and it is now impracticable to interpret these data. A radical change is needed; some steps in this process must be automated.

Peptides are molecules that play an important role in the immune system and in cell signaling. Their favorable properties make them prime candidates for initiating the design of new drugs and assist in the design of vaccines. In addition, modern synthesis techniques can quickly generate these molecules at low cost.

Statistical learning algorithms are well suited to manage large amount of data and to learn models in an automated fashion. These methods and peptides thus offer a solution of choice to the challenges facing pharmaceutical research.

We propose a kernel for learning statistical models of biochemical phenomena involving peptides. This allows, among other things, to learn a universal model that can reasonably quantify the binding energy between any peptide sequence and any binding site of a protein. In addition, it unifies the theory of many existing string kernels while maintaining a low computational complexity.

This kernel is particularly suitable for quantifying the interaction between antigens and proteins of the major histocompatibility complex. We provide a tool to predict peptides that are likely to be processed by the antigen presentation pathway. This tool has won an international competition and has several applications in immunology, including vaccine design.

Ultimately, a peptide should maximize the interaction with a target protein or maximize bioactivity in the host. We formalize this problem as a structured prediction problem. Then, we propose an algorithm exploiting the longest paths in a graph to identify peptides maximizing the predicted bioactivity of a previously learned model. We validate this new approach in the laboratory with the discovery of new antimicrobial peptides. Finally, we provide PAC-Bayes bound for two structured prediction algorithms, one of which is new.

Table des matières

Liste des tableaux

Liste des figures

Remerciements

Je remercie mon directeur de recherche, Mario Marchand, pour m'avoir donné l'opportunité de travailler sous sa supervision. Je suis énormément reconnaissant pour le support financier que Mario m'a généreusement offert. Mario est une personne passionnée, intelligente, débordante d'idées et exigeante, et je le remercie pour ces qualités. J'ai effectué ma maîtrise et mon doctorat sous sa supervision ; c'est sans hésitation que je referais ce choix.

Je remercie mon co-directeur de recherche, Jacques Corbeil pour m'avoir permis de découvrir le monde fascinant de la bioinformatique. Jacques est un chercheur exceptionnel d'une ouverture remarquable. Il a su guider mes recherches et me confronter à des problèmes inspirants. Jacques est toujours prêt aux changements et ne connaît pas la peur de l'échec. Je le remercie de m'avoir poussé à l'extérieur de ma zone de confort.

Je remercie mes parents, André et Lucie, pour leur support inconditionnel. Ils ont toujours mis leurs enfants et l'éducation en priorité. Ce sont des personnes admirables, je les remercie pour avoir été des modèles exceptionnels. Je remercie mon père pour m'avoir mis en contact en jeune âge avec l'informatique et ma mère pour m'avoir encouragé à apprendre l'anglais.

Je remercie ma conjointe, Maryse, de m'avoir accompagné dans cette aventure. Je suis privilégié de connaître et partager ma vie avec cette femme déterminée, attentionnée, généreuse et humaine. Je la remercie de m'avoir encouragé durant les moments difficiles et d'avoir célébré mes réussites. Je la remercie également d'avoir corrigé cette thèse.

Je remercie le professeur François Laviolette de m'avoir transmis sa passion des mathématiques. François est doté d'une capacité didactique rarement vue. Il est également excessivement généreux de son temps et de sa personne. Je le remercie comme collaborateur, comme professeur et pour avoir généreusement contribuer au financement de mon doctorat.

Je remercie tous les étudiants qui furent de passage au GRAAL durant ma maîtrise et mon doctorat. Spécifiquement, merci à Alexandre Drouin, Pascal Germain, Jean-Françis Roy, Alexandre Lacoste, Amélie Rolland, Alexandre Lacasse et Francis Turgeon-Boutin. Collectivement, vous avez créé une ambiance de recherche dynamique. Vos connaissances respectives et votre générosité intellectuelle ont contribué à cette thèse.

Je remercie les membres du laboratoire de Jacques pour avoir si souvent répondu à mes interrogations. Spécifiquement, merci à Sébastien Boisvert pour ses discussions et à Lynda Robitaille pour sa disponibilité.

Je remercie les collaborateurs qui ont participé à certaines des contributions scientifiques de cette thèse : Sylvain Moineau et Denise Tremblay pour les tests en cultures, Éric Biron et Xinxia Liang pour la synthèse des peptides et Claude-Guy Quimper pour son expertise en optimisation combinatoire.

Finalement, je remercie le CRSNG et le FRQNT pour la mise place des programmes de subventions qui m'ont permis de faire cette thèse.

Avant-propos

Nous avons sélectionné quatre publications que nous présenterons dans cette thèse. Ces publications se distinguent par un fil conducteur qui nous a permis, grâce à leur contribution respective, de proposer une méthode de conception de peptides pharmaceutiques assistée par algorithmes d'apprentissage automatique. Voici pour chacun, les renseignements sur les auteurs ainsi que leurs rôles dans la préparation.

Publications présentées

Learning a Peptide-Protein Binding Affinity Predictor with Kernel Ridge Regression

Cet article fut publié en 2013 dans le journal *BMC Bioinformatics* par Sébastien Giguère, Mario Marchand, François Laviolette, Alexandre Drouin et Jacques Corbeil. Sébastien Giguère est l'auteur principal de ce travail. Celui-ci est responsable de la conception du noyau, des deux algorithmes pour en faire le calcul, de la conception des expérimentations et de la majorité des résultats empiriques. Au moment de la publication, l'étudiant Alexandre Drouin était stagiaire sous la supervision de Sébastien Giguère et des professeurs Mario Marchand, François Laviolette et Jacques Corbeil. Celui-ci est responsable des résultats empiriques sur les complexes majeurs d'histocompatibilité. Les professeurs Mario Marchand, François Laviolette et Jacques Corbeil ont tous participé activement au processus de réflexion qui a mené à cette contribution ainsi qu'à l'écriture de cet article.

MHC-NP : Predicting Peptides Naturally Processed by the MHC

Cet article fut publié en 2013 dans le *Journal of Immunological Methods* par Sébastien Giguère, Alexandre Drouin, Alexandre Lacoste, Mario Marchand, Jacques Corbeil et François Laviolette. Sébastien Giguère est l'auteur principal de ce travail. Celui-ci est responsable de l'analyse du problème, de l'analyse des données et de la conception de la méthode. Au moment du développement de la méthode, l'étudiant Alexandre Drouin était stagiaire sous la supervision de Sébastien Giguère et des professeurs Mario Marchand, François Laviolette et Jacques Corbeil. Alexandre Drouin a participé aux expérimentations et au développement de

l'implémentation actuellement disponible. L'étudiant Alexandre Lacoste a participé aux expérimentations en utilisant une variante de la méthode où l'algorithme d'apprentissage utilisé avait été substitué par ses récents travaux. Cette variante n'est pas présentée dans cet article. Les professeurs Mario Marchand, François Laviolette et Jacques Corbeil ont tous participé au processus de réflexion qui a mené à cette contribution ainsi qu'à l'écriture de cet article.

Risk Bounds and Learning Algorithms for the Regression Approach to Structured Output Prediction

Cet article fut publié en 2013 lors de l'*International Conference on Machine Learning (ICML)* par Sébastien Giguère, François Laviolette, Mario Marchand et Khadidja Sylla. La version que nous présentons au Chapitre 8 est celle qui se retrouve dans le chapitre "*PAC-Bayesian Risk Bounds and Learning Algorithms for the Regression Approach to Structured Output Prediction*" du livre "*Advanced Structured Prediction*" publié par le MIT Press en 2014. Les auteurs de ce chapitre de livre sont Sébastien Giguère, François Laviolette, Mario Marchand et Amélie Rolland. Cette contribution est le fruit d'une collaboration entre le professeur Mario Marchand et l'étudiant Sébastien Giguère. Le professeur Marchand fut responsable des bornes PAC-Bayes alors que Sébastien Giguère fut responsable de l'analyse et de l'implémentation des algorithmes ainsi que des résultats présentés. Le professeur Laviolette fut un collaborateur important grâce à son expertise en théorie PAC-Bayes. Finalement, l'étudiante Amélie Rolland fut responsable d'une partie des résultats empiriques de sélection de modèle par minimisation du risque quadratique qui sont uniquement présents dans le livre *Advanced Structured Prediction*.

Machine Learning Assisted Design of Highly Active Peptides for Drug Discovery

Cet article fut soumis en 2014 au journal *PLOS Computational Biology* par Sébastien Giguère, François Laviolette, Mario Marchand, Denise Tremblay, Sylvain Moineau, Éric Biron, Xinxia Liang et Jacques Corbeil. Sébastien Giguère est l'auteur principal de ce travail. Celui-ci est responsable de la conception de tous les algorithmes, de l'implémentation de ceux-ci, des résultats et de l'écriture de l'article. Denise Tremblay et le professeur Sylvain Moineau sont responsables des tests de cultures bactériennes. L'étudiante Xinxia Liang et le professeur Éric Biron sont responsables de la synthèse et de la purification des peptides. Les professeurs Mario Marchand, François Laviolette et Jacques Corbeil ont tous participé activement au processus de réflexion qui a mené à cette contribution ainsi qu'à l'écriture de cet article.

Autres publications

Un cinquième article est mis en annexe de cette thèse. Cet article ne fait pas partie des contributions principales de cette thèse mais mérite mention. Nous croyons que dans un futur proche, la méthode présentée pourrait s'appliquer à la découverte de biomarqueurs ou faire partie de méthodes rapides pour le diagnostique en clinique.

A Pseudo-Boolean Set Covering Machine

Cet article fut présenté à la 18e édition de l'*International Conference on Principles and Practice of Constraint Programming* par Pascal Germain, Sébastien Giguère, Jean-Francis Roy, Brice Zirakiza, François Laviolette et Claude-Guy Quimper. L'article est actuellement publié dans le livre *International Conference on Principles and Practice of Constraint Programming* édité par *Springer*. Le titre de premier auteur est partagé par les quatre premiers auteurs. Ceux-ci ont contribué également à l'analyse, à la conception de la méthode, à l'implémentation et à générer les résultats présentés. Les professeurs François Laviolette et Claude-Guy Quimper ont supervisé le projet et révisé l'article.

Introduction

La découverte de nouveaux médicaments est un processus comportant plusieurs étapes complexes pouvant prendre jusqu'à 10 ans et coûter plus d'un milliard de dollars. Dans le cas du développement rationnel de médicaments, la première étape consiste à déterminer au niveau moléculaire l'entité biologique responsable de l'infection ou de la pathologie. Une enzyme ou, de manière plus générale, une protéine sera ensuite identifiée. On voudra ainsi arrêter ou stimuler l'activité biologique de cette nouvelle cible pharmaceutique.

C'est ici que s'insère cette thèse. Nous cherchons des algorithmes capables de construire une molécule à même de contrôler les activités biologiques de la cible pharmaceutique d'intérêt. Pour ce faire, plusieurs techniques existent déjà. Par exemple, le criblage à haut débit permet de filtrer une banque de molécules ayant un potentiel pharmaceutique. Cette technique possède plusieurs désavantages. D'abord, elle est très coûteuse, puisqu'on doit se procurer un éventail de molécules commercialement offertes et tester chacune d'elles empiriquement en laboratoire. Ensuite, la taille de l'espace chimique des molécules concevables est si grande (de l'ordre de 10^{60} molécules) que le criblage à haut débit permet d'explorer seulement une infime partie de cet espace.

Depuis quelques années, de nouvelles technologies comme le criblage virtuel permettent de faire une présélection des molécules candidates et de retenir seulement celles ayant une forte probabilité d'être actives. Il est ainsi possible de diminuer les coûts, d'augmenter les chances de succès et de concentrer ses efforts sur de meilleurs candidats. Il est donc envisageable de filtrer un éventail beaucoup plus large de molécules, augmentant ainsi la probabilité de découvrir un ligand significativement différent de ceux déjà découverts et ayant des propriétés exceptionnelles.

Au cours des dernières années, nos activités de recherches furent orientées vers l'apprentissage automatique de modèles permettant de faire de tel criblage. Plus précisément, nous nous sommes intéressés aux interactions entre peptides et protéines pour la conception de composés pharmaceutiques et à l'étude d'antigènes pour la conception de vaccins. Cette thèse fut au centre d'un projet de recherche interdisciplinaire composé de biologistes, de chimistes, d'informaticiens, de mathématiciens et d'experts en génomique qui ont travaillé en collaboration sur le projet ambitieux de liaison entre peptides et protéines.

Les résultats préliminaires à cette thèse, qui furent obtenus lors de la maîtrise de Sébastien Giguère, ont permis à notre groupe de recherche d'obtenir une subvention du Fonds de recherche du Québec - Nature et technologies pour le projet en équipe (FRQNT-Équipe) intitulé "Analyse de liaisons entre peptides et protéines cibles par le biais de méthodes combinatoires et d'apprentissage machine dans le but d'aider à la découverte de composés pharmaceutiques". Cette subvention de $180,000$\$ sur une période de 3 ans a permis de subventionner mon doctorat et d'initier plusieurs projets et collaborations. Le succès et les retombées de cette subvention ont récemment permis à notre groupe d'obtenir une importante subvention de $1,5$ M\$ octroyée par le Consortium québécois sur la découverte du médicament (CQDM-FOCUS) pour le projet intitulé "Approches computationelles et apprentissage machine pour la génération de peptides bioactifs facilitant la découverte de médicaments". Cette subvention permettra à notre groupe de continuer à explorer certaines des idées présentées dans cette thèse.

Cette thèse se divise en deux sections. La première section présente les concepts de base nécessaires à la compréhension des articles. Ainsi, le Chapitre 1 présente les notions de biochimie et de pharmacologie et le Chapitre 2 présente les notions d'apprentissage automatique. Nous considérons que la revue de littérature faite dans chacun des articles présentés est suffisante puisqu'elle cible spécifiquement les travaux connexes à chaque contribution. Pour cette raison, nous concentrons nos efforts dans les deux premiers chapitres à introduire les concepts préalables pour qu'un lecteur moins initié puisse apprécier cette thèse plutôt que de refaire une revue de littérature.

On retrouve dans la seconde section de cette thèse les quatre articles qui la composent. Au Chapitre 4, nous présentons un nouveau noyau pour faire l'apprentissage de modèles d'interactions entre peptides et protéines. Nous démontrons que l'utilisation de ce nouveau noyau permet d'obtenir des modèles d'interactions dont la précision surpasse l'état de l'art pour plusieurs problèmes biochimiques.

Au Chapitre 6, nous présentons un outil permettant de prédire les antigènes présentés en surface des cellules par les complexes majeurs d'histocompatibilité. Cet outil fait usage du noyau présenté au Chapitre 4 et fait suite à une compétition organisée par le *Dana-Farber Cancer Institute* lors de laquelle cet outil a été déterminé le plus précis des méthodes participantes. Ces avancements permettront, entre autres de mieux comprendre certaines maladies auto-immunes et d'avancer la recherche des vaccins de prochaine génération, ceux à base d'épitopes.

Au Chapitre 8, nous présentons deux bornes de type PAC-Bayes sur le risque de prédiction de deux algorithmes de prédiction de structures. Nous retrouvons un algorithme déjà proposé en minimisant la première borne alors que la seconde borne suggère un nouvel algorithme. Les deux algorithmes sont basés sur le principe de minimisation du risque quadratique plutôt que le risque de prédiction. Cette caractéristique permet l'apprentissage d'un modèle beaucoup plus rapidement. Ces algorithmes, en contraste avec les algorithmes de régression ou de clas-

sification, permettent de prédire des structures complexes comme des bio-molécules. Bien que le champ d'application de ces algorithmes ne soit pas limité à la bioinformatique, ils offrent une nouvelle approche à des problèmes comme le séquençage de peptides par spectrométrie de masse ou la prédiction de la structure secondaire de protéines. Cette meilleure compréhension des algorithmes de prédiction de structure nous a permis de nous rapprocher du but ultime de cette thèse, soit prédire des peptides à haute bioactivité.

Au Chapitre 10, nous proposons un algorithme pour rapidement identifier la séquence peptidique maximisant la fonction d'inférence des méthodes à noyaux, lorsque le noyau est celui présenté au Chapitre 4. À titre de preuve de concept, nous avons appris un modèle de l'activité anti-microbienne de peptides. Nous utilisons ce modèle et l'algorithme présenté dans ce chapitre pour identifier les meilleurs candidats peptidiques. Certains de ces candidats sont synthétisés et leur activité anti-microbienne est mesurée en laboratoire contre deux bactéries : *Escherichia coli* et *Staphylococcus aureus*. La majorité des peptides ont démontré une activité anti-microbienne et un des candidats synthétisés possède une activité dépassant les meilleurs peptides connus. Cette preuve de concept démontre non seulement l'efficacité de cette nouvelle approche pour découvrir des antibiotiques, mais aussi que la méthodologie peut s'appliquer aux premières étapes du développement de tout médicament à base peptidique.

Chapitre 1

Notions de biochimie et de pharmacologie

Cette section a pour objectif de fournir les notions de biochimie, de biologie et de pharmacologie nécessaires à la compréhension des contributions de cette thèse. Il ne s'agit en rien d'un substitut à un ouvrage de référence puisque nous présenterons uniquement les notions préalables aux articles présentés. Un lecteur initié peut possiblement sauter cette section alors qu'un non-initié pourra mieux apprécier les contributions de cette thèse s'il en fait la lecture.

1.1 Les peptides

Chez les eucaryotes, c'est-à-dire les cellules à noyau, on retrouve 20 acides aminés qui sont encodés par leur contenu génétique. Ceux-ci sont les pièces élémentaires qui servent à construire les protéines, à la base du fonctionnement cellulaire. Un peptide est un polymère résultant de la liaison de plusieurs acides aminés par un lien peptidique. Tout comme les protéines, les acides aminés sont liés deux à deux par un lien covalent entre la fonction carboxyle de l'un et la fonction amine de l'autre. On distingue les peptides des protéines simplement par leur nombre d'acides aminés, les peptides en ayant moins de 30.

Les peptides peuvent facilement être synthétisés en laboratoire. En effet, plusieurs laboratoires sont maintenant équipés de synthétiseurs de peptides. Ces machines accélèrent et automatisent la synthèse, diminuant ainsi les coûts associés à la production de ceux-ci.

Les peptides sont naturellement présents dans le corps humain et sont importants dans la régulation de plusieurs mécanismes biologiques comme les interactions entre protéines et le fonctionnement du système immunitaire. De plus, la simplicité de synthèse des peptides en fait des outils intéressants pour interroger la biologie lors d'expérimentations. Finalement, comme nous le verrons, ils peuvent être utilisés pour la conception de médicaments ou de vaccins. Pour toutes ces raisons, ces molécules auront une place importante dans cette thèse.

1.1.1 Les peptides en tant que matrices et outils d'investigation

Un inhibiteur peptidique est un peptide qui inhibe la fonction d'une protéine, altérant potentiellement certaines voies de signalisation cellulaire et, par le fait même, le comportement de la cellule. Ces inhibiteurs peuvent être utilisés pour disséquer le rôle d'une protéine en biologie des systèmes, ou encore, valider une cible pharmaceutique. Un peptide peut ensuite être utilisé comme composé précurseur en chimie médicinale pour le développement de nouveaux inhibiteurs d'application thérapeutique ou diagnostique.

Par contre, il est très rare qu'un peptide devienne un médicament : celui-ci doit pénétrer dans la cellule, être stable dans le corps humain, posséder des propriétés pharmacocinétiques favorables, etc. Effectivement, la grande majorité des peptides sont dégradés très rapidement par l'organisme. Cependant, ceux-ci peuvent être améliorés et transformés en médicaments par des techniques de peptidomimétique [Kharb et al., 2011].

Un peptidomimétique est une petite molécule utilisée pour imiter un peptide. Certaines modifications à la structure moléculaire d'un peptide permettent de l'imiter tout en augmentant la stabilité ou l'activité biologique. Ces modifications incluent la cyclisation du peptide, l'alkylation, la pégylation ou la conjugaison du peptide à une molécule de transport. Une autre approche consiste à remplacer les acides aminés de type L par des acides aminés de type D. Ceux-ci ne sont pas naturellement produits chez les eucaryotes et sont utilisés pour leurs propriétés thérapeutiques. Ils peuvent être pris par voie orale et sont efficaces pendant une période de temps plus longue grâce à leur plus grande résistance à la digestion enzymatique [Latacz et al., 2006]. Les peptidomimétiques offrent donc plusieurs avantages par rapport aux peptides natifs. Hruby et al [Hruby and Cai, 2013] ont rapporté quelques-uns des avantages évidents de l'utilisation de ces molécules dans le développement de nouveaux médicaments : une stabilité accrue à la protéolyse, un meilleur transport dans la membrane cellulaire, une biodisponibilité supérieure et une affinité accrue pour les interactions entre protéines [Latacz et al., 2006]. Finalement, ces petites molécules naturelles n'ont peu ou pas de toxicité, une propriété souhaitable.

Nous proposons au Chapitre 10 une méthode assistée par apprentissage automatique permettant la conception de peptides pouvant être transformé en médicament par des techniques de peptidomimétique.

1.2 Rôle des peptides dans le système immunitaire

Le système immunitaire a été introduit au grand public principalement par l'entremise des campagnes de vaccination. Par contre, une meilleure connaissance du système immunitaire ne permettra pas uniquement de faire des vaccins contre les pathogènes émergeants, mais de meilleurs vaccins qui offriront une protection accrue. Par exemple, à chaque année, le vaccin

contre l'*influenza* ne protège pas la totalité des individus vaccinés.

Chaque personne possède un système immunitaire très particulier et l'arrangement des gènes du système immunitaire est expressément fait pour maximiser la diversité. De cette façon, un nouveau pathogène a très peu de chances d'infecter la totalité d'une espèce, maximisant ainsi l'espérance de survie de celle-ci. Malheureusement, ce mécanisme astucieux complexifie également la conception d'un vaccin puisque celui-ci doit protéger une population hétérogène. Aussi, ce réarrangement des gènes peut parfois causer un dérèglement du système immunitaire à la base des maladies auto-immunes comme la sclérose en plaques, le diabète de type 1, le lupus, la polyarthrite rhumatoïde ou la maladie de Crohn. Finalement, la singularité du système immunitaire est à la base du rejet d'un organe transplanté chez son receveur. Celui-ci possède un système immunitaire qui lui est si spécifique qu'il attaque l'organe de son donneur.

Récemment, il a été démontré que les cellules mutantes à la base de certains cancers se retrouvent, en fait, chez la majorité des individus. Ce qui semble anormal chez les patients souffrant d'un cancer serait l'incapacité de leur système immunitaire de détruire ces cellules anomales [de Visser et al., 2006]. Une meilleure connaissance de ce système pourrait ouvrir de nouvelles voies thérapeutiques. Déjà, le Québec a lancé une campagne de vaccination contre le virus du papillome humain (VPH) dans l'objectif de prévenir le cancer du col de l'utérus chez les femmes.

L'immunologie est un domaine de recherche excessivement vaste et complexe. Dans le cadre de cette thèse, nous nous limiterons au processus de reconnaissance des antigènes par les complexes majeurs d'histocompatibilité. Ce processus de liaison est central au déclenchement de la réponse immunitaire.

1.2.1 Complexes majeurs d'histocompatibilité

Un antigène est une substance, très souvent un peptide, qui provoque une réponse du système immunitaire acquis. L'antigène provient typiquement de l'extérieur du corps, par exemple d'une protéine appartenant à une bactérie à l'origine d'une infection.

Les complexes majeurs d'histocompatibilité (MHC) sont une famille de protéines polymorphiques codées par le locus HLA du sixième chromosome. Ceux-ci sont responsables de la reconnaissance spécifique de l'antigène et du déclenchement de la réponse immunitaire cellulaire. Les molécules du MHC font partie d'une voie de signalisation complexe, responsable de la présentation des antigènes à la surface des cellules présentatrices d'antigène. Les antigènes sont présentés sous la forme de complexes MHC-peptide, où le peptide est un fragment de la séquence protéique de l'antigène. Une fois le complexe MHC-peptide présenté à la surface de la cellule présentatrice d'antigène, les lymphocytes T reconnaissent spécifiquement certains complexes et déclenchent la réponse immunitaire appropriée. Les molécules du MHC sont classées en deux catégories : les MHC de classe I présentent des antigènes principalement aux

lymphocytes T auxiliaires (CD8) alors que les MHC de classe II présentent des antigènes principalement aux lymphocytes T cytotoxiques (CD4). Finalement, les deux voies sont complexes et la liaison d'un peptide à un MHC peut être vérifiée en laboratoire ou par des méthodes computationnelles. Nous présentons au Chapitre 4 une approche par apprentissage automatique pour la prédiction des interactions entre peptides et protéines avec laquelle nous avons obtenu d'excellents résultats pour la prédiction des complexes MHC-peptide.

Un épitope est la partie de l'antigène qui est reconnue par le système immunitaire. Les vaccins à base de ceux-ci sont très prometteurs pour les maladies pour lesquelles les approches actuelles, comme l'atténuation de pathogènes, ne sont pas efficaces ou facilement réalisables. Ces vaccins ont l'avantage d'être plus simple à produire et d'induire une réponse immunitaire très spécifique en ciblant la région immunogénique d'un antigène (épitope) et un MHC spécifiques à l'hôte [Toussaint and Kohlbacher, 2009]. Le développement de méthodes assistées par ordinateur pour l'identification des épitopes immunogènes représente une étape importante pour faciliter la création de ces vaccins de prochaine génération. Idéalement, ceux-ci pourraient être adaptés pour cibler des sous-groupes de la population, comme les femmes enceintes ou les personnes immunosupprimées, pour qui les vaccins à base de pathogènes atténués présentent des risques plus importants.

La méthode que nous présentons au Chapitre 6 est un premier pas dans cette direction. Elle permet de prédire les antigènes qui survivront le processus de présentation cellulaire. En effet, pour produire une réponse immunitaire, un antigène doit se lier au MHC mais sa liaison n'est pas garante d'une réponse immunitaire. La méthode que nous présentons a été déterminée la plus précise lors d'une compétition organisée par le *Dana-Farber Cancer Institute* en 2012.

1.3 Les interactions entre protéines comme cible pharmaceutique

Les conséquences des maladies génétiques conduisent généralement à un dysfonctionnement cellulaire impliquant les interactions entre protéines. Puisque les peptides et les protéines sont les molécules ayant le plus grand impact dans la signalisation cellulaire, les peptides sont donc particulièrement bien adaptés pour régénérer les activités cellulaires dysfonctionnelles.

Cibler les interactions protéine-protéine augmente donc considérablement la variété et la quantité des cibles pharmaceutiques et aura un impact énorme à la fois sur le choix des cibles et sur les effets secondaires du médicament. L'interaction d'un ligand avec une protéine cible est basée sur la reconnaissance moléculaire entre un motif de liaison et un domaine d'interaction de la protéine. Compte tenu de la nature de la surface d'interaction, la capacité d'une molécule synthétique à imiter la structure secondaire des protéines est cruciale pour lier efficacement le domaine d'interaction et moduler l'activité de celle-ci [Mullard, 2012, Fry, 2006, Wells and

McClendon, 2007]. Grâce à leur capacité d'imiter la structure secondaire des protéines, les peptides représentent donc un modèle de choix [Sillerud and Larson, 2005, Jesus Perez de Vega et al., 2007]. Dans de rares cas, il est même possible d'utiliser un peptide pour stabiliser un complexe protéique afin de régénérer l'activité cellulaire dysfonctionnelle. Il est important de mentionner qu'à ce jour, les petites molécules de moins de 500 daltons, en contraste aux peptides (1,000 daltons et plus), sont prédominantes dans la recherche d'inhibiteurs.

1.4 Criblage *in-vitro* à haut débit

Les méthodes de criblage à haut débit utilisent des instruments robotisés, des systèmes de manipulation de fluides, plusieurs types de détecteurs et des logiciels pour permettre aux scientifiques de rapidement accomplir des millions de tests biologiques. Ces méthodes accélèrent considérablement la recherche de composés chimiques qui pourront servir de point de départ pour la conception d'un nouveau médicament. D'abord, nous présenterons quelques approches pour mesurer l'activité biologique d'une molécule. Puis, nous présenterons deux approches de criblage à haut débit.

1.4.1 Mesure de l'activité d'une molécule

La méthode utilisée pour mesurer l'activité d'une molécule varie énormément selon ce que celle-ci doit accomplir et la compréhension du phénomène étudié. De plus, certaines limitations expérimentales nécessitent parfois que l'activité d'une molécule soit mesurée indirectement. L'activité moléculaire peut être divisée en deux grandes familles : la mesure de la liaison avec une autre molécule et la mesure d'un changement phénotypique (caractère observable).

On mesure la liaison entre deux molécules dans l'objectif de déterminer l'énergie nécessaire pour séparer celles-ci. La constante de dissociation est couramment utilisée en pharmacologie. Celle-ci est donnée par

$$K_d \stackrel{\text{def}}{=} \frac{[P][L]}{[PL]}\,, \tag{1.1}$$

où $[P], [L]$ et $[PL]$ sont respectivement la concentration molaire de la protéine, du ligand et du complexe protéine-ligand à l'équilibre. La constante de dissociation s'exprime donc en unités molaires. Lorsque $[P] = K_d$, on a donc que $\frac{[PL]}{[L]+[PL]} = \frac{1}{2}$ puisque $[L] = [PL]$. Ce qui s'interprète comme la concentration de ligand $[L]$ nécessaire pour que le site de liaison de la protéine $[P]$ soit à moitié occupé. Plus la constante de dissociation est petite, plus le ligand est fortement lié à la protéine. Une forte liaison moléculaire est typiquement reliée à une forte bio-activité, mais n'est pas garante de celle-ci.

On mesure le changement phénotypique ou la bio-activité d'une molécule dans l'objectif de déterminer si celle-ci accomplit bien la tâche attendue. Par exemple, on peut mesurer l'impact d'une molécule sur la croissance cellulaire, la taille des cellules, le taux de mortalité ou

l'espérance de vie d'animaux, la grosseur des tumeurs, etc. En variant la concentration de la molécule, il est possible de déterminer l'efficacité de celle-ci. La concentration inhibitrice médiane (IC_{50}) est une mesure de l'efficacité d'une substance dans l'inhibition d'une fonction biologique ou biochimique spécifique. Elle indique la concentration nécessaire pour stopper 50% d'un processus biologique donné. Similairement, la concentration minimale inhibitrice (MIC) est utilisée pour tester l'efficacité des antibiotiques ou la résistance de certaines souches bactériennes à ceux-ci. Elle correspond à la plus petite concentration stoppant la croissance bactérienne. En contraste, la métrique LD_{50} permet de mesurer la toxicité d'une molécule en déterminant la concentration léthale chez 50% des individus d'une étude.

1.4.2 Criblage par chimiothèques

Une chimiothèque est une banque de molécules, souvent disponible commercialement, qui peut contenir jusqu'à plusieurs millions de composés chimiques. On dit qu'une chimiothèque est focalisée lorsque ses composés partagent certaines propriétés ou qu'ils appartiennent à une même famille. Il existe également des chimiothèques de composés pharmaceutiques déjà approuvés par les agences gouvernementales. Celles-ci sont utilisées dans l'espoir de trouver une nouvelle application aux composés pharmaceutiques déjà approuvés. Malheureusement, ce type de chimiothèque est sur-utilisé par les compagnies pharmaceutiques, ce qui freine énormément la découverte de nouvelles molécules.

Le criblage par chimiothèque est pratiquement toujours robotisé. Ces robots testent, à plusieurs concentrations, tous les composés de la chimiothèque. La réaction est faite sur des plaques qui contiennent plusieurs petits puits dans lesquels le test biologique est effectué. Une lecture, typiquement optique, permet de quantifier l'issue de l'expérimentation.

1.4.3 Criblage peptidique par chimie combinatoire

Cette technologie permet de fabriquer rapidement, en laboratoire, une chimiothèque combinatoire contenant un très grand nombre de peptides différents. Celle-ci a été utilisée avec succès pour découvrir des ligands pour des récepteurs [Kumaresan et al., 2011, Liu et al., 2009], des protéines [Alluri et al., 2003, Joo and Pei, 2008, Martínez-Ceron et al., 2011, Zhang et al., 2008] et des facteurs de transcription [Liu et al., 2011, Alluri et al., 2006].

Pour ce faire, 20 récipients $(r_1, \ldots, r_{20})$ sont utilisés, soit un récipient par acide aminé $(a_1, \ldots, a_{20})$. Tous les récipients contiennent des microbilles sur lesquelles les peptides sont synthétisés. Pour construire des peptides de n acides aminés, on exécute le protocole suivant :

1. Lors de la première itération, l'acide aminé a_i est couplé aux billes du récipient r_i. Pour les itérations subséquentes, a_i est couplé à la fin du dernier acide aminé ajouté.
2. Les 20 récipients sont transvidés dans un même gros récipient R qu'on agite pour mélanger les billes.

3. Le contenu de R est séparé en 20 quantités égales qu'on transvide dans $(r_1, \ldots, r_{20})$.
4. Répéter n fois les étapes 1 à 3.

Comme les peptides ont généralement moins de 30 acides aminés, moins de 30 d'itérations sont nécessaires pour construire un échantillon des 20^{30} peptides possibles. Lorsque la librairie de peptides est synthétisée, les étapes suivantes permettent d'identifier les peptides qui ont une affinité avec la protéine cible.

1. Une molécule phosphorescente est attachée à la protéine cible.
2. Plusieurs instances de la protéine phosphorylée sont ajoutées aux billes accueillant les peptides. Ceux ayant une affinité avec la protéine s'y lieront.
3. Le substrat est lavé pour enlever les protéines qui ne se sont pas liées.
4. Les billes accueillant un peptide ayant une affinité avec la protéine (phosphorylée) seront phosphorescentes. Celles-ci sont récupérées à l'aide d'un microscope ou d'un autre procédé automatisé.
5. Les peptides sont séparés de la protéine et de la bille.
6. Les peptides sont séquencés par spectrométrie de masse.
7. Les peptides sont resynthétisés pour en obtenir une plus grande quantité.
8. Les peptides sont purifiés, par exemple, par chromatographie en phase liquide à haute performance.
9. La constante de dissociation (K_d) entre la protéine et le peptide est mesurée en utilisant des technologies comme l'interférométrie optique par nanopores [Latterich and Corbeil, 2008].

De façon alternative, la protéine cible peut être couplée à la biotine. La sélection se fait alors par affinité entre la biotine et la streptavidine, une liaison très forte permettant cette sélection. Bien que la chimie combinatoire soit une excellente méthode pour synthétiser un large éventail de molécules, le nombre de n-peptides possibles (20^n) étant beaucoup trop élevé, on en aura seulement un infime échantillon. En effet, six millions de peptides générés aléatoirement de 9 résidus couvrent une fraction infime (moins de $0,0016\%$) de l'ensemble des 20^9 peptides. En conséquence, il est pratiquement certain que les meilleurs peptides de grande taille ne seront pas présents dans la librairie. Nous proposons au Chapitre 10 de cette thèse une approche basée sur des algorithmes d'apprentissage automatique qui permet d'attaquer astucieusement ce problème combinatoire.

1.5 Criblage *in-silico* à haut débit

Les approches *in-silico* sont celles qui ont recours à un modèle mathématique pour prédire s'il y aura interaction entre une molécule et une protéine. Typiquement, ce type d'approche

est moins précise que les approches *in-vitro* mais elles ont l'avantage d'être plus rapides et beaucoup moins dispendieuses.

En améliorant la précision de ces méthodes, il sera possible de remplacer une plus grande partie des expérimentations faites en laboratoire, diminuant considérablement les coûts. À ce jour, le criblage virtuel est utilisé principalement pour faire un premier filtre des molécules candidates. La méthode décrite Chapitre 10 est à même de proposer des composés chimiques. Ceux-ci furent testés en laboratoire avec succès, démontrant ainsi le progrès de ce type de méthode.

1.5.1 Dynamique moléculaire

Dans le domaine de la modélisation moléculaire, le *docking* est une méthode qui permet de prédire l'orientation préférentielle d'une molécule par rapport à une seconde lorsqu'ils forment un complexe stable. Cette méthode peut être considérée comme un problème de serrure et de clés, où l'on intéresse à trouver la bonne orientation de la clé (ligand) qui ouvre la serrure (protéine).

Le *docking* doit donc résoudre un problème d'optimisation, qui cherche virtuellement le meilleur ajustement d'un ligand dans la pochette de liaison d'une protéine. Le but est donc de parvenir à optimiser la conformation et l'orientation relative de la protéine et du ligand, de telle sorte que l'énergie libre du système soit réduite au minimum. Une fois l'ajustement préférentiel du ligand déterminé, il est possible de déterminer l'énergie de liaison du complexe à l'aide d'un modèle de champs de forces [Huey et al., 2007].

Toutefois, puisque le ligand et la protéine sont flexibles, il est plus juste de voir cette analogie comme celle d'insérer une main dans un gant [Jorgensen, 1991]. Au cours du processus, le ligand et la protéine ajustent leurs conformations jusqu'à parvenir au meilleur ajustement. Ce genre d'ajustement conformationnel se nomme l'ajustement induit [Wei et al., 2004].

Malheureusement, déterminer cette conformation est particulièrement difficile. La cristallographie aux rayons X nous permet uniquement de connaître la structure d'une protéine avec et sans son ligand dans la conformation qui a été cristallisée. Ces informations donnent peu d'informations sur les mécanismes d'ajustements et la conformation qu'aura la protéine avec d'autres ligands. Finalement, cette technique est particulièrement coûteuse en temps de calcul, ce qui rend trop difficile le filtrage de larges banques de molécules.

1.5.2 Apprentissage statistique d'un modèle

En contraste avec les méthodes de dynamique moléculaire où le modèle utilisé est défini par un ensemble d'experts, l'approche par apprentissage de modèles se base sur des cas empiriques (exemples d'apprentissage) pour apprendre un modèle ou un ensemble de règles. Le modèle appris peut ensuite être utilisé pour déterminer s'il y aura interaction entre une molécule et

une protéine. Il s'agit de l'approche que nous préconiserons dans cette thèse. Nous présenterons dans le prochain chapitre comment de tels modèles peuvent être appris. Nous verrons également comment cette approche peut permettre de rapidement filtrer de larges banques de molécules.

Chapitre 2

Notions d'apprentissage automatique

2.1 Apprentissage supervisé

L'apprentissage supervisé est une tâche de l'apprentissage automatique qui consiste à apprendre le modèle d'un phénomène à partir d'exemples d'apprentissage. Ce chapitre présente donc les bases de cette science.

2.1.1 Exemples d'apprentissage

En apprentissage supervisé, l'algorithme d'apprentissage a accès à un ensemble d'apprentissage $S \stackrel{\text{def}}{=} \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$ de m exemples où chaque exemple est une paire $(\mathbf{x}, \mathbf{y})$, où $\mathbf{y}$ est la sortie attendue lorsque l'entrée est $\mathbf{x}$. On dénote par $\mathcal{X}$ l'ensemble des entrées possibles et par $\mathcal{Y}$ l'ensemble des sorties possibles, ainsi $S \subseteq \mathcal{X} \times \mathcal{Y}$. De plus, les exemples d'apprentissage dans S sont générés indépendamment d'une distribution inconnue D. La tâche de l'algorithme d'apprentissage est donc de produire une fonction $h : \mathcal{X} \to \mathcal{Y}$.

Initialement, l'apprentissage automatique s'est intéressé à apprendre des fonctions $h : \mathbb{R}^n \to \mathbb{R}$ où l'entrée devait être un vecteur et la sortie un nombre réel. Depuis quelques années, la venue des noyaux permet, entre autres, d'apprendre de telles fonctions lorsque $\mathcal{X}$ et $\mathcal{Y}$ sont des structures arbitraires. Les bio-molécules se représentant difficilement sous forme de vecteurs, ces derniers avancements ont multiplié les applications de l'apprentissage automatique en bio-informatique.

2.1.2 Procédure d'apprentissage

La fonction de perte $\ell(h(\mathbf{x}), \mathbf{y})$ permet de quantifier l'erreur du prédicteur h sur l'exemple $\mathbf{x}$ alors que la sortie souhaitée est $\mathbf{y}$. On a toujours que $\ell(h(\mathbf{x}), \mathbf{y}) = 0$ lorsque $h(\mathbf{x}) = \mathbf{y}$. Cependant, la valeur de la fonction de perte lorsque $h(\mathbf{x}) \neq \mathbf{y}$ dépend de la tâche d'apprentissage et de la structure de $\mathbf{y}$. Par exemple, si $\mathbf{x}$ est une molécule et que $\mathbf{y} \in \{+1, -1\}$ représente la toxicité de cette molécule où $+1$ signifie toxique et -1 non toxique, on aura simplement

$\ell(h(\mathbf{x}), \mathbf{y}) = 1$ lorsque $h(\mathbf{x}) \neq \mathbf{y}$. Nous verrons d'autres exemples de fonctions de perte dans les prochaines sections.

L'hypothèse principale en apprentissage automatique est que chaque exemple est tiré selon une distribution inconnue D. La tâche de l'algorithme d'apprentissage est donc de trouver le prédicteur h ayant le plus petit risque $R(h)$ défini comme la perte espérée

$$R(h) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},\mathbf{y})\sim D} \ell(h(\mathbf{x}), \mathbf{y})\,. \tag{2.1}$$

Par contre, l'algorithme d'apprentissage n'a pas accès à la distribution D. En revanche, il a accès à un ensemble d'apprentissage $S = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$ de m exemples tirés de D. Le risque empirique est donc la perte moyenne du prédicteur h sur les exemples d'apprentissage :

$$R_S(h) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{(\mathbf{x},\mathbf{y})\in S} \ell(h(\mathbf{x}), \mathbf{y})\,. \tag{2.2}$$

Plusieurs stratégies existent pour apprendre un modèle à partir des exemples d'apprentissage, nous en verrons quelques-unes dans cette thèse. Les stratégies que nous présenterons varient selon la tâche d'apprentissage, mais auront l'avantage d'être invariant par rapport à la structure des exemples d'apprentissage. Dans tous les cas, un second ensemble d'exemples T, qu'on nomme ensemble de test, est tiré de la même distribution D et est utilisé pour estimer $R(h)$. Cet ensemble permet donc d'estimer de la précision du prédicteur h sur les futurs exemples à venir et de comparer les différents algorithmes d'apprentissage.

2.1.3 Validation croisée

La validation croisée est une technique utilisée pour estimer la précision qu'aura un algorithme sur les futurs exemples non-observés sans utiliser l'ensemble de test T. Elle peut permettre d'éviter le problème de sur-apprentissage caractéristique d'un modèle ayant un faible risque empirique mais un risque élevé sur les futurs exemples à venir. De plus, la majorité des algorithmes d'apprentissage ont des paramètres à ajuster et la validation croisée permet, entre autres, de fixer ces paramètres.

La procédure consiste à séparer aléatoirement l'ensemble d'apprentissage S en n sous-ensembles disjoints. De ces sous-ensembles, un sera utilisé comme ensemble de validation, alors que les $n-1$ autres seront utilisés pour faire l'apprentissage du modèle. Ainsi, en répétant n fois cette procédure, il est possible d'obtenir une estimation de la performance de l'algorithme d'apprentissage, qu'on nomme risque de validation croisée. Pour ce faire, on calcule les métriques désirées sur l'union des n sous-ensembles de validation. Cette estimation est faiblement biaisée, mais heureusement, ce biais est rarement problématique en pratique. Souvent, nous utilisons $n = 10$ et lorsque n est égal au nombre d'exemples d'apprentissage, on dit qu'il s'agit d'une validation croisée de type *leave-one-out*.

Ainsi, lorsqu'on doit ajuster les paramètres d'un algorithme, les éléments d'une grille de combinaisons seront testés sur l'ensemble d'apprentissage et le tuple de paramètres ayant le plus faible risque de validation croisée sera sélectionné. L'algorithme d'apprentissage sera ensuite exécuté avec ces paramètres sur la totalité de S. Finalement, en prédisant la sortie attendue des exemples dans T, nous obtiendrons une estimation non biaisée de $R(h)$.

2.2 Noyaux

La majorité des algorithmes d'apprentissage sont basés sur des principes d'algèbre linéaire. Pour cette raison, les exemples d'apprentissage doivent être représentés par des vecteurs d'attributs. Malheureusement, les structures avec lesquelles nous travaillons en biologie sont plutôt des chaînes de caractères (acides aminés, ADN) ou des structures atomiques complexes (composés chimiques, protéines cristallisées). Les noyaux (*kernel* en anglais) permettent de remplacer la représentation vectorielle des exemples par la similarité avec les autres exemples d'apprentissage.

Pour ce faire, un noyau $k_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ calcule le produit scalaire entre deux exemples $\mathbf{x}$ et $\mathbf{x}'$ en les projetant implicitement dans un espace de caractéristiques $\mathcal{H}_{\mathcal{X}}$. L'espace de caractéristiques est un espace vectoriel à haute dimensionnalité ou, de façon plus générale, un espace de Hilbert à noyau reproduisant (*reproducing kernel Hilbert space*, RKHS). Ainsi la fonction de projection $\phi_{\mathcal{X}}$ est définie comme $\phi_{\mathcal{X}} : \mathcal{X} \to \mathcal{H}_{\mathcal{X}}$ et le noyau est défini comme

$$k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') \stackrel{\text{def}}{=} \langle \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}), \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}') \rangle \,, \tag{2.3}$$

où $\langle \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}), \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}') \rangle$ dénote le produit scalaire dans $\mathcal{H}_{\mathcal{X}}$. Cette projection est dite implicite parce que le calcul du produit scalaire est typiquement moins coûteux que la projection $\phi_{\mathcal{X}}$. Cette astuce algorithmique est à l'origine de son nom, soit le *kernel trick*. Puisque $k_{\mathcal{X}}$ calcule un produit scalaire dans l'espace de caractéristiques $\mathcal{H}_{\mathcal{X}}$, on doit prouver que $k_{\mathcal{X}}$ est positif semi-défini si l'espace de caractéristiques ne peut être explicitement décrit. Il est parfois souhaitable d'avoir $0 \leq k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') \leq 1$ pour tout $\mathbf{x}$ et $\mathbf{x}'$. Pour ce faire, tout noyau peut être normalisé ainsi

$$\frac{k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}')}{\sqrt{k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}) k_{\mathcal{X}}(\mathbf{x}', \mathbf{x}')}} \,. \tag{2.4}$$

Dans ce cas, le vecteur $\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})$ sera de norme unitaire pour tout $\mathbf{x}$ et on aura toujours que $k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}) = 1$. Finalement, les algorithmes d'apprentissage pour lesquels il est possible de remplacer les opérations d'algèbre linéaire par une expression du noyau font partie des méthodes à noyaux.

2.2.1 Noyau Spectrum (n-gram)

À titre d'exemple, le noyau Spectrum (ou n-gram) est parmi les noyaux pour chaînes de caractères les plus simples. Étant donné deux chaînes $\mathbf{x} = x_1, \ldots, x_{|\mathbf{x}|}$ et $\mathbf{x}' = x'_1, \ldots, x'_{|\mathbf{x}'|}$

d'un alphabet $\mathcal{A}$, la similarité est donnée par le nombre de sous-chaînes de taille n que $\mathbf{x}$ et $\mathbf{x}'$ ont en commun. Ainsi, le noyau Spectrum est défini par

$$k_s(n, \mathbf{x}, \mathbf{x}') \stackrel{\text{def}}{=} \sum_{i=0}^{|\mathbf{x}|-n} \sum_{j=0}^{|\mathbf{x}'|-n} I\left(x_{[i+1]}, \ldots, x_{[i+n]} = x'_{[j+1]}, \ldots, x'_{[j+n]}\right) , \tag{2.5}$$

où I est la fonction indicatrice et vaut 1 lorsque les sous-chaînes sont identiques, 0 autrement. Une propriété intéressante de ce noyau est qu'il permet la comparaison de chaînes de taille différente. L'espace de caractéristiques implicitement induit par le noyau Spectrum est $\mathcal{A}^n$, soit l'ensemble de toutes les chaînes de taille n de l'alphabet $\mathcal{A}$. Finalement, remarquons que la complexité de calcul de ce noyau dépend de la taille des chaînes $\mathbf{x}$ et $\mathbf{x}'$, en contraste avec la taille de l'espace de caractéristiques $|\mathcal{A}|^n$.

Nous présenterons au Chapitre 4 de cette thèse un noyau spécialement adapté aux petites séquences d'acides aminés comme les peptides. Nous démontrerons que ce noyau est bien positif semi-défini et que son utilisation permet de prédire la bioactivité de peptides, les interactions entre peptides et protéines et les antigènes présentés par les complexes majeurs d'histocompatibilité, tous avec une précision surpassant l'état de l'art.

2.2.2 Matrice de gram

La matrice de gram ou *kernel matrix* est une matrice symétrique carrée qui contient le produit scalaire entre tous les couples d'exemples de l'ensemble d'apprentissage :

$$K_{i,j} \stackrel{\text{def}}{=} k_{\mathcal{X}}(\mathbf{x}_i, \mathbf{x}_j) : i, j \in \{1, \ldots, m\} . \tag{2.6}$$

Les valeurs de noyaux étant au coeur des algorithmes à noyaux, la fonction qui en fait le calcul est inévitablement très sollicitée. L'objectif premier de cette matrice est donc de conserver ces valeurs pour en éviter le recalcul. Nous verrons que plusieurs algorithmes d'apprentissage à noyaux expriment leur problème d'optimisation à même cette matrice. Pour cette raison, la matrice $\mathbf{K}$, tout comme la fonction de noyau $k_{\mathcal{X}}$, doit être symétrique positive semi-définie (SPSD) : on aura que $\mathbf{K}$ est SPSD si et seulement si il existe une transformation $\phi_{\mathcal{X}}$ tel que $k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}')$ est le produit scalaire entre $\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})$ et $\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}')$. On peut vérifier que la matrice $\mathbf{K}$ est positive semi-définie si

$$\mathbf{v}^\top \mathbf{K} \mathbf{v} \geq 0 \tag{2.7}$$

pour tout $\mathbf{v} \in \mathbb{R}^m$.

2.3 La classification binaire

Le problème de classification consiste à assigner une classe à un exemple $\mathbf{x}$. Pour la classification binaire, les deux classes possibles sont $+1$ et -1, respectivement la classe positive et la

classe négative. Ainsi, les exemples d'apprentissage consistent en couples $(\mathbf{x}, y) \in \mathcal{X} \times \{+1, -1\}$. Le problème d'apprentissage consiste donc à apprendre une fonction $h : \mathcal{X} \to \{+1, -1\}$.

Un problème de classification connu en bioinformatique est la prédiction des interactions entre molécules. Dans ce cas, pour une molécule cible (possiblement une protéine), l'ensemble d'apprentissage sera constitué de molécules se liant à la cible $(+1)$ et d'autres ne se liant pas (-1).

Les machines à vecteurs de support [Cortes and Vapnik, 1995] sont parmi les meilleurs algorithmes de classification à noyaux. Nous présenterons brièvement leur fonctionnement au Chapitre 6. Puis, nous expliquerons comment elles ont été utilisées en combinaison avec un noyau spécialisé pour les peptides afin de prédire les antigènes naturellement présentés par les complexes majeurs d'histocompatibilité. Nous avons utilisé l'approche du Chapitre 6 lors de la *Machine Learning Competition in Immunology* organisée par le *Dana-Farber Cancer Institute* en 2012. Nous avons remporté le premier prix de cette compétition pour la méthode la plus précise.

2.3.1 Métriques pour la classification

Il existe plusieurs métriques pour mesurer la capacité d'un classificateur à correctement prédire la classe de nouveaux exemples. On dénotera par P le nombre d'exemples de classe positive, par TP (*True Positive*) le nombre d'exemples correctement classifiés comme positifs et par FP (*False Positive*) les exemples de classe négative incorrectement classifiés positifs. Similairement, on dénote par N le nombre d'exemples de la classe négative, TN (*True Negative*) le nombre d'exemples correctement classifiés négatifs et FN (*False Negative*) le nombre d'exemples incorrectement classifiés comme négatifs.

Ainsi, la sensibilité est la fraction des exemples positifs correctement classifiés :

$$\text{sensibilité} = \frac{TP}{P} \,. \tag{2.8}$$

Similairement, la spécificité est la fraction des exemples négatifs correctement classifiés :

$$\text{spécificité} = \frac{TN}{N} \,. \tag{2.9}$$

Le risque d'un classificateur est la fraction des exemples incorrectement classifiés :

$$\text{risque} = \frac{FP + FN}{P + N} \,. \tag{2.10}$$

La précision d'un classificateur est la fraction des exemples correctement prédits comme positifs sur le nombre total d'exemples prédits positifs :

$$\text{précision} = \frac{TP}{(TP + FP)} \,. \tag{2.11}$$

Finalement, la F-mesure est une moyenne harmonique entre la précision et la sensibilité :

$$\text{F-mesure} = 2 \cdot \frac{\text{précision} \cdot \text{sensibilité}}{\text{précision} + \text{sensibilité}} . \tag{2.12}$$

2.4 La régression

La régression consiste à assigner à chaque exemple $\mathbf{x}$ un nombre réel. Ainsi, les exemples d'apprentissage consistent en couples $(\mathbf{x}, y) \in \mathcal{X} \times \mathbb{R}$. La tâche d'apprentissage est donc d'apprendre une fonction $h : \mathcal{X} \to \mathbb{R}$.

La régression est particulièrement bien adaptée aux problèmes de biochimie et aux interractions entre molécules. En effet, il est souvent possible de quantifier la bioactivité d'une molécule ou de mesurer la force d'interaction entre celle-ci et une protéine. Ainsi, un régresseur permet d'obtenir une prédiction plus informative et fine du phénomène étudié.

La régression de Ridge à noyaux est un algorithme de régression que nous avons préconisé pour plusieurs raisons lors de cette thèse. Les motivations derrière ce choix ainsi que les détails de l'algorithme seront présentés dès le Chapitre 4. De plus, nous proposons une garantie de performance de type PAC-Bayes au Chapitre 8 qui est également valide pour cet algorithme.

2.4.1 Métriques pour la régression

La fonction de perte quadratique $\ell(h(\mathbf{x}), y) = (h(\mathbf{x}) - y)^2$ est certainement la plus adaptée à la régression. Celle-ci possède l'avantage d'être positive, continue et convexe, ce qui mène généralement à des problèmes d'optimisation relativement faciles à résoudre. Ainsi, l'erreur quadratique moyenne (*mean square error*, MSE) est une métrique intuitive :

$$MSE = \frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x}_i) - y_i\right)^2 . \tag{2.13}$$

Malheureusement, cette métrique ne possède pas les même unités que la valeur prédite, ce qui la rend difficilement interprétable. Pour cette raison, il est courant d'en prendre la racine carrée, ainsi le *root mean square error* (RMSE) est défini par

$$RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x}_i) - y_i\right)^2} . \tag{2.14}$$

Ces dernières métriques sont difficilement comparables puisque leurs valeurs dépendent de l'échelle de $\mathcal{Y}$. Par exemple, supposons que la tâche d'apprentissage est de prédire la taille d'un objet. Selon qu'on exprime la taille en centimètres ou en mètres, les valeurs de MSE et RMSE changeront, alors que la précision et les prédictions du prédicteur restent les mêmes. Afin de faciliter l'interprétation et la comparaison entre problèmes d'apprentissage, il est souhaitable

d'utiliser le coefficient de détermination :

$$R^2 = 1 - \frac{\sum_{i=1}^{m} \left(h(\mathbf{x}_i) - y_i\right)^2}{\sum_{i=1}^{m} \left(\bar{y} - y_i\right)^2}, \tag{2.15}$$

où $\bar{y} = \frac{1}{m} \sum_{i=1}^{m} y_i$. En plus d'être comprise entre 0 et 1, cette métrique est particulièrement bien adaptée à la régression en raison de sa relation directe avec la perte quadratique.

Pour des fins de comparaison avec certaines méthodes existantes, nous utiliserons parfois le coefficient de corrélation ou *Pearson's r*. Cette métrique indique le niveau de corrélation entre les valeurs prédites et les valeurs attendues. Un coefficient de corrélation de 0 indique aucune corrélation. Une valeur de $+1$ indique une corrélation parfaite : $h(\mathbf{x}_i) = y_i$ pour tout i. En contraste, une valeur de -1 indique une corrélation parfaite, mais inverse : $h(\mathbf{x}_i) = -y_i$ pour tout i. Ainsi le Pearson's r est défini comme

$$r = \frac{\sum_{i=1}^{m}(h(\mathbf{x}_i) - \bar{h})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m}(h(\mathbf{x}_i) - \bar{h})^2}\sqrt{\sum_{i=1}^{m}(y_i - \bar{y})^2}}, \tag{2.16}$$

où $\bar{h} = \frac{1}{m} \sum_{i=1}^{m} h(\mathbf{x}_i)$ et $\bar{y} = \frac{1}{m} \sum_{i=1}^{m} y_i$.

Finalement, notons que contrairement au coefficient de détermination R^2, nous pouvons avoir un coefficient de corrélation $r = 1$ même lorsque les prédictions $h(\mathbf{x}_i)$ sont très mauvaises. Cela ce produit lorsque $h(\mathbf{x}_i) = ay_i + b$ pour tout i car, dans ce cas, on a $r = 1$ pour tout choix de a et b. Par contre, si on a un prédicteur h avec $r = 1$, il est possible (et facile) de trouver a et b tel que $ah(\mathbf{x}) + b$ nous donne un prédicteur parfait ayant $ah(\mathbf{x}_i) + b = y_i$ pour tout i.

2.5 La prédiction de structures

La prédiction de structures est la tâche d'apprentissage la plus générale. L'entrée et la sortie des exemples d'apprentissage $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ sont tous deux des structures complexes. Grâce aux noyaux, autant l'entrée que la sortie d'un prédicteur peut être une séquence, un arbre, un graphe, une protéine, un peptide, la structure d'une molécule, etc. La tâche d'apprentissage est donc d'apprendre une fonction $h : \mathcal{X} \to \mathcal{Y}$. Par exemple, $\mathbf{x}$ peut être un mot manuscrit numérisé, donc une matrice de pixels, et $\mathbf{y}$ la séquence de caractères de ce mot. Dans ce cas, on parle de reconnaissance de mot, une tâche similaire, mais plus complexe, que la classification de caractères. Il pourrait être possible d'apprendre un traducteur si $\mathbf{x}$ était une phrase dans la langue d'origine et $\mathbf{y}$ la même phrase traduite dans la langue de destination. Au Chapitre 10 de cette thèse, nous présenterons un prédicteur où l'entrée $\mathbf{x}$ est une protéine et la sortie $\mathbf{y}$ est le peptide ayant la plus grande affinité avec la protéine $\mathbf{x}$. Nous utiliserons cette approche pour découvrir de nouveaux peptides capables de stopper la croissance bactérienne de bactéries infectieuses comme *Escherichia coli* et *Staphylococcus aureus*.

2.5.1 Le problème de pré-image

Rappelons que les noyaux calculent un produit scalaire dans un espace de caractéristiques à haute dimensionalité. Les algorithmes de prédiction de structures nécessitant deux noyaux, nous ferons donc la distinction entre $k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}')$, le noyau sur l'entrée, et $k_{\mathcal{Y}}(\mathbf{y}, \mathbf{y}')$, le noyau sur la sortie. Dénotons par $\phi_{\mathcal{X}}$ et $\phi_{\mathcal{Y}}$ les fonctions qui projettent les éléments de $\mathcal{X}$ (l'ensemble des entrées possibles) et $\mathcal{Y}$ (l'ensemble des sorties possibles) respectivement vers les espaces de caractéristiques $\mathcal{H}_{\mathcal{X}}$ et $\mathcal{H}_{\mathcal{Y}}$. Ainsi, nous avons $\phi_{\mathcal{X}} : \mathcal{X} \to \mathcal{H}_{\mathcal{X}}$ et $\phi_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{H}_{\mathcal{Y}}$.

Afin de rendre les algorithmes de prédiction de structures à noyaux indépendants de la structure du problème, la tâche d'apprentissage consiste à apprendre d'abord un opérateur linéaire $\mathbf{W} : \mathcal{H}_{\mathcal{X}} \to \mathcal{H}_{\mathcal{Y}}$ qui projette les éléments de l'espace de caractéristiques d'entrée $\mathcal{H}_{\mathcal{X}}$ vers $\mathcal{H}_{\mathcal{Y}}$, l'espace de caractéristiques de la structure de sortie. Il existe plusieurs approches pour apprendre un tel opérateur linéaire ; nous en présenterons deux au Chapitre 8. La première fut proposée par Cortes et al. [2007] alors que la deuxième est nouvelle. Pour les deux, nous fournissons des garanties de performance de type PAC-Bayes.

Ainsi, la fonction de prédiction h est donnée par

$$h(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} \left\| \boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y}) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}) \right\| , \tag{2.17}$$

où $\|\cdot\|$ représente la norme euclidienne. Ce problème de minimisation est connu sous le nom de pré-image. Il consiste à trouver la structure $\mathbf{y}$ parmi l'ensemble de toutes les structures possibles $\mathcal{Y}$, ayant les caractéristiques $\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y})$ les plus proches des caractéristiques prédites $\mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})$. Ce problème combinatoire est presque toujours NP-Difficile [Gärtner and Vembu, 2009]. Heureusement, dans certains cas, il est possible de le résoudre. Nous en verrons quelques-uns au Chapitre 8.

2.5.2 Métriques pour la prédiction de structures

Le choix de la métrique en prédiction de structures dépend, évidemment, de la tâche d'apprentissage et de la structure. Il est courant d'utiliser des mesures de distance, mais d'autres métriques ne respectant pas l'inégalité du triangle peuvent également être utilisées.

Une métrique évidente qui fonctionne pour toutes les structures est la perte $\{0, 1\}$ qui compte une perte de 0 si la structure est exactement la bonne, 1 autrement. Malheureusement, cette métrique est souvent trop restrictive. Pour la prédiction de mot, il est possible d'utiliser la distance de Hamming, ce qui correspond à compter le nombre de lettres incorrectement prédites. Pour les problèmes de bioinformatique où il peut se produire des insertions ou des délétions de caractères, il est préférable d'utiliser une métrique moins sensible à ce type d'erreur, comme la distance de Levenshtein.

En somme, l'expert du domaine est souvent le mieux placé pour déterminer une métrique adéquate pour évaluer les prédictions.

Chapitre 3

Présentation du premier article

3.1 Détails de l'article

Titre :	Learning a Peptide-Protein Binding Affinity Predictor with Kernel Ridge Regression
Auteurs :	Sébastien Giguère, Mario Marchand, François Laviolette, Alexandre Drouin, Jacques Corbeil
Lieu de publication :	BMC Bioinformatics
Type de publication :	Article journal
Année :	2013

3.2 Contexte

Les résultats préliminaires à cet article ont permis à notre groupe de recherche d'obtenir une subvention du Fond québécois de recherche en nature et technologies pour le projet en équipe (FQRNT-Équipe) intitulé "Analyse de liaisons entre peptides et protéines cibles par le biais de méthodes combinatoires et d'apprentissage machine dans le but d'aider à la découverte de composés pharmaceutiques". Cette subvention de $180,000$\$ sur une période de 3 ans a permis de subventionner cette thèse et d'initier plusieurs projets et collaborations.

3.3 Contributions et discussion

La principale contribution de cette publication est le *Generic String kernel* (GS). Il s'agit d'un noyau spécialement conçu pour les petites séquences d'acides aminés comme les peptides et les pseudo-séquences d'interfaces de liaison. Ce noyau a la particularité d'unifier plusieurs noyaux conçus pour différents problèmes de bioinformatique. En effet, huit noyaux précédemment proposés sont des cas particuliers du *Generic String*. Nous avons développé un algorithme par programmation dynamique pour calculer rapidement la valeur de celui-ci. Nous fournissons également la preuve que le noyau proposé satisfait la condition de Mercer, prouvant ainsi qu'il

est un noyau valide. De plus, certaines propriétés du noyau permettent d'en faire rapidement l'approximation. Il est rare de pouvoir faire une telle approximation puisque les noyaux pour séquences sont typiquement discret.

Grâce à celui-ci, à de récents avancements dans les noyaux pour protéines cristallisées et à la régression de ridge nous avons réussi à apprendre un prédicteur d'affinités de liaison universel. Celui-ci est capable de prédire, avec une précision raisonnable, l'affinité de liaison entre tout couple peptide-protéine, à condition que la structure cristallisée de la protéine soit connue. Bien que des prédicteurs similaires pour les petites molécules pharmaceutiques existaient déjà, par exemple pour les récepteurs couplés aux protéines G [Jacob et al., 2008], le prédicteur que nous proposons se distingue pour deux raisons. Premièrement, il n'est pas restreint à une famille de protéine ; en théorie, il peut être utilisé pour cribler n'importe quelle protéine. Aussi, l'ensemble des peptides possibles est plusieurs ordres de grandeur plus grand que l'ensemble des petites molécules pharmaceutiques commercialement disponibles. Pour cette raison, il est d'autant plus important de s'outiller d'une méthode rapide et peu coûteuse pour en faire le criblage virtuellement plutôt qu'en laboratoire.

Empiriquement, nous démontrons que ce noyau surpasse l'état de l'art sur plusieurs problèmes biochimiques reliés aux peptides. Entre autre, le *Generic String* semble particulièrement bien adapté pour prédire les antigènes se liant aux complexes majeurs d'histocompatibilité. L'ensemble d'entraînement dépassant $14,000$ exemples, nous avons dû porter une attention particulière à l'implémentation du noyau, à la régression de ridge ainsi qu'à la validation croisée. Les grappes de calcul fournies par Calcul Canada ont contribué à la réussite des expérimentations sur ces molécules.

Finalement, nous démontrons empiriquement que la régression de ridge à noyaux (*kernel ridge regression*, KRR) semble préférable à la régression par vecteurs de support (*support vector regression*, SVR) pour la prédiction de l'affinité de liaison entre peptides et protéines.

Chapitre 4

Learning a peptide-protein binding affinity predictor with kernel ridge regression

The cellular function of a vast majority of proteins is performed through physical interactions with other biomolecules, which, most of the time, are other proteins. Peptides represent templates of choice for mimicking a secondary structure in order to modulate protein-protein interaction. They are thus an interesting class of therapeutics since they also display strong activity, high selectivity, low toxicity and few drug-drug interactions. Furthermore, predicting peptides that would bind to a specific MHC alleles would be of tremendous benefit to improve vaccine based therapy and possibly generate antibodies with greater affinity. Modern computational methods have the potential to accelerate and lower the cost of drug and vaccine discovery by selecting potential compounds for testing in silico prior to biological validation.

We propose a specialized string kernel for small bio-molecules, peptides and pseudo-sequences of binding interfaces. The kernel incorporates physico-chemical properties of amino acids and elegantly generalize eight kernels, comprised of the Oligo, the Weighted Degree, the Blended Spectrum, and the Radial Basis Function. We provide a low complexity dynamic programming algorithm for the exact computation of the kernel and a linear time algorithm for it's approximation. Combined with kernel ridge regression and SupCK, a novel binding pocket kernel, the proposed kernel yields biologically relevant and good prediction accuracy on the PepX database. For the first time, a machine learning predictor is capable of predicting the binding affinity of any peptide to any protein with reasonable accuracy. The method was also applied to both single-target and pan-specific Major Histocompatibility Complex class II benchmark datasets and three Quantitative Structure Affinity Model benchmark datasets.

On all benchmarks, our method significantly (p-value $\leq$ 0.057) outperforms the current state-of-the-art methods at predicting peptide-protein binding affinities. The proposed ap-

proach is flexible and can be applied to predict any quantitative biological activity. Moreover, generating reliable peptide-protein binding affinities will also improve system biology modelling of interaction pathways. Lastly, the method should be of value to a large segment of the research community with the potential to accelerate the discovery of peptide-based drugs and facilitate vaccine development. The proposed kernel is freely available at `http://graal.ift.ulaval.ca/downloads/gs-kernel/`.

4.1 Background

The cellular function of a vast majority of proteins is performed through physical interactions with other proteins. Indeed, essentially all of the known cellular and biological processes depend, at some level, on protein-protein interactions (PPI) [Toogood, 2002, Albert, 2005]. Therefore, the controlled interference of PPI with chemical compounds provides tremendous potential for the discovery of novel molecular tools to improve our understanding of biochemical pathways as well as the development of new therapeutic agents [Wells and McClendon, 2007, Dömling, 2008].

Considering the nature of the interaction surface, protein secondary structures are essential for binding specifically to protein interaction domains. Peptides also represent templates of choice for mimicking a secondary structure in order to modulate protein-protein interactions [Costantino and Barlocco, 2006, Jesus Perez de Vega et al., 2007]. Furthermore, they are a very interesting class of therapeutics since they display strong activity, high selectivity, low toxicity and fewer drug-drug interactions. They can also serve as investigative tools to gain insight into the role of a protein, by binding to distinct regulatory regions to inhibit specific functions.

Yearly, large sums of money are invested in the process of finding druggable targets and identifying compounds with medicinal utility. The widespread use of combinatorial chemistry and high-throughput screening in the pharmaceutical and biotechnology industries implies that millions of compounds can be tested for biological activity. However, screening large chemical libraries generates significant rates of both false positives and negatives. The process is expensive and faces a number of challenges in testing candidate drugs and validating the hits, all of which must be done efficiently to reduce costs and time. Computational methods with reasonable predictive power can now be envisaged to accelerate the process providing an increase in productivity at a reduced cost.

As an example, peptides ranging from 8 to 12 AA represent the recognition unit for the MHC (Major Hiscompatibility Complex). Being capable of predicting which peptides bind to a specific MHC alleles would be of tremendous benefit to improve vaccine based therapy, possibly generating antibodies with greater affinity that could yield an improved immune response. Moreover, simply having data on the binding affinity of peptides and proteins could

readily assist system biology modelling of interaction pathways.

The ultimate goal is to build a predictor of the highest binding affinity peptides. This task would be facilitated if one had a fast and accurate binding affinity predictor. Indeed, with this predictor, one could easily predict the binding affinity of huge sets of peptides and select the candidates with the highest predicted binding affinity, or use stochastic search methods such as simulated annealing if the set of peptides were too large. This paper provides a step in this direction with the use of a machine learning algorithm based on kernel methods and a novel kernel.

Traditional machine learning approaches focused on using binary binding data for classification of compounds (binding, non-binding) [Jacob et al., 2008, Jacob and Vert, 2008]. Non-binding compounds are rarely known and valuable quantitative binding affinity information is lost during training, a major obstacle to binary classification. Other approaches used information from the US Food and Drug Administration's adverse event reporting system for the prediction of off-target protein interactions [Takarabe et al., 2012]. These methods can predict unknown drug-target interactions from FDA approved drugs but are not suited for the identification of new pharmaceutical compounds. New databases, such as the PepX database, contain binding affinities between peptides and a large group of protein families. The first part of this paper presents a general method for learning a binding affinity predictor between any peptide and any protein, a novel machine learning contribution to biology.

The Immune Epitope Database (IEDB) [Peters et al., 2005] contains a large number of binding affinities between peptides and Major Histocompatibility Complex (MHC) alleles. Predicting methods for MHC class I alleles have already obtained great success [Zhang et al., 2012, Jacob and Vert, 2008]. The simpler binding interface of MHC-I molecules makes the learning problem significantly easier than for MHC-II molecules. Allele specific (single-target) methods for MHC class II alleles have also reasonable accuracy, despite requiring a large number of training examples for every allele in order to achieve adequate accuracy [Zhang et al., 2012]. Pan-specific (multi-target) methods, such as MultiRTA[Bordner and Mittelmann, 2010a] and NetMHCIIpan-2.0[Nielsen et al., 2010], were designed in order to overcome this issue. These methods can predict, with reasonable accuracy, the binding affinity of a peptide to any MHC allele, even if this allele has no known peptide binders.

We propose a new machine learning approach based on kernel methods [Shawe-Taylor and Cristianini, 2004] capable of both single-target and multi-target (pan-specific) prediction. We searched for kernels that encode relevant binding information for both proteins and peptides. Therefore, we propose a new kernel, a Generic String (GS) kernel, that generalizes most of kernels currently used in this setting (RBF [Shawe-Taylor and Cristianini, 2004], Blended spectrum [Shawe-Taylor and Cristianini, 2004], Oligo [Meinicke et al., 2004], Weighted Degree [Rätsch and Sonnenburg, 2004], ...). The GS kernel is shown to be a suitable similarity

measure between peptides and pseudo-sequences of MHC-II binding interfaces.

For the machine learning algorithm itself, we show that kernel ridge regression [Shawe-Taylor and Cristianini, 2004] (KRR) is generally preferable to the support vector regression (SVR) algorithm [Smola and Schölkopf, 2004] because KRR has less hyperparameters to tune than SVR, thus making the learning time smaller. The regression score obtained with the PepX examples is competitive with the ones generated on data sets containing peptides binding to a single protein, even if the former task is, in theory, much more difficult. For the peptide-MHC binding problem, comparison on benchmark datasets show that our algorithm surpasses NetMHCIIpan-2.0[Nielsen et al., 2010], the current state-of-the-art method. Indeed, in the most difficult pan-specific case (when the algorithm is trained on all alleles except the allele used for testing), our algorithm performs better than the state of the art in most cases. Finally, we have found that ridge regression outperforms SVR on three quantitative structure affinity model (QSAM) single-target predictions benchmarks [Zhou et al., 2010]. We thus propose a machine learning approach to immunology and a novel string kernel which have shown to yield impressive results on benchmark datasets for various biological problems.

4.2 Methods

4.2.1 Statistical machine learning and kernel ridge regression in our context

Given a set of training examples (or cases), the task of a learning algorithm is to build an accurate predictor. In this paper, each example will be of the form $((\mathbf{x}, \mathbf{y}), e)$, where $\mathbf{x}$ represents a peptide, $\mathbf{y}$ represents a protein, and e is a real number representing the binding energy (or the binding affinity) between the peptide $\mathbf{x}$ and the protein $\mathbf{y}$. A multi-target predictor is a function h that returns an output $h(\mathbf{x}, \mathbf{y})$ when given any input $(\mathbf{x}, \mathbf{y})$. In our setting, the output $h(\mathbf{x}, \mathbf{y})$ is a real number estimate of the "true" binding energy (or the binding affinity) e between $\mathbf{x}$ and $\mathbf{y}$. The predictor h is accurate on example $((\mathbf{x}, \mathbf{y}), e)$ if the predicted output $h(\mathbf{x}, \mathbf{y})$ is very similar to the real output e. A predictor is good when it is accurate on most future examples unseen during training.

With kernel methods, each input $(\mathbf{x}, \mathbf{y})$ is implicitly mapped to a *feature vector* $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) = (\phi_1(\mathbf{x}, \mathbf{y}), \phi_2(\mathbf{x}, \mathbf{y}), \ldots, \phi_d(\mathbf{x}, \mathbf{y}))$ of large dimensionality d. Moreover, the predictor is represented by a real-valued weight vector $\mathbf{w}$ that lies in the space of feature vectors. Given an arbitrary input $(\mathbf{x}, \mathbf{y})$, the output of the predictor $h_{\mathbf{w}}$ is given by the scalar product

$$h_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \sum_{i=1}^{d} w_i \phi_i(\mathbf{x}, \mathbf{y}) \,.$$

The lost incurred by predicting a binding energy $h_{\mathbf{w}}(\mathbf{x}, \mathbf{y})$ on input $(\mathbf{x}, \mathbf{y})$, when the true binding energy is e, is measured by a *loss function* $\ell(\mathbf{w}, (\mathbf{x}, \mathbf{y}), e)$. As is usual in regression, we

will use the quadratic loss function

$$\ell(\mathbf{w}, (\mathbf{x}, \mathbf{y}), e) = (e - \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}))^2 .$$

The fundamental assumption in machine learning is that each example $((\mathbf{x}, \mathbf{y}), e)$ is drawn according to some unknown distribution D. Then the task of the learning algorithm is to find the predictor $h_{\mathbf{w}}$ having the smallest possible *risk* $R(h_{\mathbf{w}})$ defined as the expected loss

$$R(h_{\mathbf{w}}) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{((\mathbf{x},\mathbf{y}),e)\sim D} \ell(\mathbf{w}, (\mathbf{x}, \mathbf{y}), e) .$$

However, the learning algorithm does not have access to D. Instead, it has access to a training set $S \stackrel{\text{def}}{=} \{((\mathbf{x}_1, \mathbf{y}_1), e_1), ((\mathbf{x}_2, \mathbf{y}_2), e_2), \ldots, ((\mathbf{x}_m, \mathbf{y}_m), e_m)\}$ of m examples where each example $((\mathbf{x}_i, \mathbf{y}_i), e_i)$ is assumed to be generated independently according to the same (but unknown) distribution D. Modern statistical learning theory [Shawe-Taylor and Cristianini, 2004, Schölkopf and Smola, 2002] tells us that the predictor $h_{\mathbf{w}}$ minimizing the *ridge regression cost function* $F(S, \mathbf{w})$ will have a small risk $R(h_{\mathbf{w}})$ whenever the obtained value of $F(S, \mathbf{w})$ is small. Here, $F(S, \mathbf{w})$ is defined as

$$F(S, \mathbf{w}) \stackrel{\text{def}}{=} \|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \ell(\mathbf{w}, (\mathbf{x}_i, \mathbf{y}_i), e_i) = \|\mathbf{w}\|^2 + C \sum_{i=1}^{m} (e_i - \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i))^2 ,$$

for some suitably-chosen constant $C > 0$. The first term of $F(S, \mathbf{w})$, $\|\mathbf{w}\|^2 \stackrel{\text{def}}{=} \mathbf{w} \cdot \mathbf{w}$, which is the squared Euclidean norm of $\mathbf{w}$, is called a *regularizer* and it penalizes predictors having a large norm (complex predictors). The second term measures the accuracy of the predictor on the training data. Consequently, the parameter C controls the complexity-accuracy trade-off. Its value is usually determined by measuring the accuracy of the predictor on a separate ("hold-out") part of the data that was not used for training, or by more elaborate sampling methods such as cross-validation.

The *representer theorem* [Shawe-Taylor and Cristianini, 2004, Schölkopf and Smola, 2002] tells us that the predictor $\mathbf{w}^*$ that minimizes $F(S, \mathbf{w})$ lies in the linear subspace span by the training examples. In other words, we can write

$$\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) ,$$

where the coefficients α_i are called the *dual* variables and provide collectively the dual representation of the predictor. This change of representation makes the cost function dependent on $\boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i)$ only via the scalar product $\boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j, \mathbf{y}_j) \stackrel{\text{def}}{=} k((\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_j, \mathbf{y}_j))$ for each pair of examples. The function k is called a *kernel* and has the property of being efficiently computable for many feature maps $\boldsymbol{\phi}$, even if the feature space induced by $\boldsymbol{\phi}$ has an extremely large dimensionality. By using k instead of $\boldsymbol{\phi}$, we can construct linear predictors in feature

spaces of extremely large dimensionality with a running time that scales only with the size of the training data (with no dependence on the dimensionality of $\boldsymbol{\phi}$). This fundamental property is also known as the *kernel trick* [Shawe-Taylor and Cristianini, 2004, Schölkopf and Smola, 2002]. It is important to point out that, since a kernel corresponds to a scalar product in a feature space, it can be considered as a similarity measure. A large (positive) value of the kernel normally implies that the corresponding feature vectors point in similar directions, although a value close to zero normally implies that the two feature vectors are mostly orthogonal (dissimilar).

As was proposed by several authors [Jacob and Vert, 2008, Nagamine and Sakakibara, 2007, Faulon et al., 2008, Jacob et al., 2008], we restrict ourselves to joint feature maps having the form $\boldsymbol{\phi}(\mathbf{x},\mathbf{y}) = \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}) \otimes \boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y})$ where $\otimes$ denotes the tensor product. The tensor product between two vectors $\mathbf{a} = (a_1,\ldots,a_n)$ and $\mathbf{b} = (b_1,\ldots,b_m)$ denotes the vector $\mathbf{a}\otimes\mathbf{b} = (a_1b_1, a_1b_2, \ldots, a_nb_m)$ of all the nm products between the components of $\mathbf{a}$ and $\mathbf{b}$. If we now define the peptide kernel $k_{\mathcal{X}}$ by $k_{\mathcal{X}}(\mathbf{x},\mathbf{x}') \stackrel{\text{def}}{=} \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})\cdot\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}')$, and the protein kernel $k_{\mathcal{Y}}$ by $k_{\mathcal{Y}}(\mathbf{y},\mathbf{y}') \stackrel{\text{def}}{=} \boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y})\cdot\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y}')$, the joint kernel k simply decomposes as the product of $k_{\mathcal{X}}$ and $k_{\mathcal{Y}}$ because

$$\begin{aligned} k((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) &\stackrel{\text{def}}{=} \boldsymbol{\phi}(\mathbf{x},\mathbf{y})\cdot\boldsymbol{\phi}(\mathbf{x}',\mathbf{y}') \\ &= \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})\otimes\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y})\cdot\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}')\otimes\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y}') \\ &= (\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})\cdot\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}'))(\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y})\cdot\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y}')) \\ &\stackrel{\text{def}}{=} k_{\mathcal{X}}(\mathbf{x},\mathbf{x}')k_{\mathcal{Y}}(\mathbf{y},\mathbf{y}')\,. \end{aligned}$$

Consequently, from the representer theorem we can write the multi-target predictor as

$$h_{\mathbf{w}^*}(\mathbf{x},\mathbf{y}) = \mathbf{w}^*\cdot\boldsymbol{\phi}(\mathbf{x},\mathbf{y}) = \mathbf{w}^*\cdot(\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})\otimes\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y})) = \sum_{i=1}^{m}\alpha_i k_{\mathcal{X}}(\mathbf{x}_i,\mathbf{x})k_{\mathcal{Y}}(\mathbf{y}_i,\mathbf{y})\,.$$

In the case of the quadratic loss $\ell(\mathbf{w},(\mathbf{x},\mathbf{y}),e) = (e-\mathbf{w}\cdot\phi(\mathbf{x},\mathbf{y}))^2$, $F(S,\mathbf{w})$ is a strongly convex function of $\mathbf{w}$ for any strictly positive C. In that case, there exists a single local minimum which coincides with the global minimum. This single minimum is given by the point $\mathbf{w}^*$ where the gradient $\partial F(S,\mathbf{w})/\partial\mathbf{w}$ vanishes. For the quadratic loss, this solution $\mathbf{w}^*$ is given by

$$\boldsymbol{\alpha} = \left(\mathbf{K} + \frac{1}{C}\mathbf{I}\right)^{-1}\mathbf{e}\,, \tag{4.1}$$

where $\boldsymbol{\alpha} \stackrel{\text{def}}{=} (\alpha_1,\ldots,\alpha_m)$, $\mathbf{e} \stackrel{\text{def}}{=} (e_1,\ldots,e_m)$, $\mathbf{K}$ denotes the Gram matrix of kernel values $K_{i,j} = k_{\mathcal{X}}(\mathbf{x}_i,\mathbf{x}_j)k_{\mathcal{Y}}(\mathbf{y}_i,\mathbf{y}_j)$, and $\mathbf{I}$ denotes de $m\times m$ identity matrix. Hence, the learning algorithm for kernel ridge regression just consists at solving Equation (4.1). Note that for a symmetric positive semi-definite kernel matrix $\mathbf{K}$, the inverse of $\mathbf{K}+\mathbf{I}/C$ always exists for any finite value of $C>0$. Note also that the inverse of an $m\times m$ matrix is obtained in $O(m^3)$ time with the Gaussian-elimination method and in $O(m^{2.376})$ time with the Coppersmith-Winograd algorithm.

Finally, we will also consider the single protein target case where only one protein y is considered. In this case, the predictor $h_{\mathbf{w}}$ predicts the binding energy from a feature vector $\boldsymbol{\phi}_{\mathcal{X}}$ constructed only from the peptide. Hence, the predicted binding energy for peptide $\mathbf{x}$ is now given $\mathbf{w} \cdot \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})$. So, in this single protein target case, the cost function to minimize is still given by $F(S, \mathbf{w})$ but with $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ replaced by $\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})$. Consequently, in this case, the solution is still given by Equation (4.1) but with a kernel matrix $\mathbf{K}$ given by $K_{i,j} = k_{\mathcal{X}}(\mathbf{x}_i, \mathbf{x}_j)$. The single-target predictor is thus given by

$$h_{\mathbf{w}^*}(\mathbf{x}) = \mathbf{w}^* \cdot \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i k_{\mathcal{X}}(\mathbf{x}_i, \mathbf{x}) \,.$$

Kernel methods have been extremely successful within the last decade, but the choice of the kernel is critical for obtaining good predictors. Hence, confronted with a new application, we must be prepared to design an appropriate kernel. The next subsections show how we have designed and chosen both peptide and protein kernels.

4.2.2 A generic string (GS) kernel for small bio-molecule strings

String kernels for bio-molecules have been applied with success in bioinformatics and computational biology. Kernels for large bio-molecules, such as the local-alignment kernel [Saigo et al., 2004] have been well studied and applied with success to problems such as protein homology detection. However, we observed that these kernels perform rather poorly on smaller compounds (data not shown). Consequently, kernels designed for smaller bio-molecules like peptides and pseudo sequences have recently been proposed. Some of these kernels [Meinicke et al., 2004] exploit sub-string position uncertainty while others [Toussaint et al., 2010] use physicochemical properties of amino acids. We present a kernel for peptides that exploits both of these properties in a unified manner.

The proposed kernel, which we call the generic string (GS) kernel, is a similarity measure defined for any pair $(\mathbf{x}, \mathbf{x}')$ of strings of amino acids. Let Σ be the set of all amino acids. Then, given any string $\mathbf{x}$ of amino acids (e.g., a peptide), let $|\mathbf{x}|$ denote the length of string $\mathbf{x}$, as measured by the number of amino acids in $\mathbf{x}$. The positions of amino acids in $\mathbf{x}$ are numbered from 1 to $|\mathbf{x}|$. In other words, $\mathbf{x} = x_1, x_2, \ldots, x_{|\mathbf{x}|}$ with all $x_i \in \Sigma$.

Now, let $\boldsymbol{\psi} : \Sigma \longrightarrow \mathbb{R}^d$ be an encoding function such that for each amino acid a,

$$\boldsymbol{\psi}(a) = (\psi_1(a), \psi_2(a), \ldots \psi_d(a)) \tag{4.2}$$

is a vector where each component $\psi_i(a)$ encodes one of the d properties (possibly physicochemical) of amino acid a. In a similar way, we define $\boldsymbol{\psi}^l : \Sigma^l \longrightarrow \mathbb{R}^{dl}$ as an encoding function for strings of length l. Thus, $\boldsymbol{\psi}^l(\mathbf{a})$ encodes all l amino acids of $\mathbf{a}$ concatenning l vectors, each of d components :

$$\boldsymbol{\psi}^l(a_1, a_2, .., a_l) \stackrel{\text{def}}{=} (\boldsymbol{\psi}(a_1), \boldsymbol{\psi}(a_2), \ldots, \boldsymbol{\psi}(a_l)) \tag{4.3}$$

Let $L \geq 1$ be a maximum length for substring comparison. We define the generic string (GS) kernel as the following similarity function over any pair $(\mathbf{x}, \mathbf{x}')$ of strings of length at least L :

$$GS(\mathbf{x}, \mathbf{x}', L, \sigma_p, \sigma_c) \stackrel{\text{def}}{=} \sum_{l=1}^{L} \sum_{i=0}^{|x|-l} \sum_{j=0}^{|x'|-l} e^{\left(\frac{-(i-j)^2}{2\sigma_p^2}\right)} e^{\left(\frac{-\|\boldsymbol{\psi}^l(x_{i+1},..,x_{i+l}) - \boldsymbol{\psi}^l(x'_{j+1},..,x_{j+l})\|^2}{2\sigma_c^2}\right)}. \tag{4.4}$$

In other words, this kernel compares each substring $x_{i+1}, x_{i+2}, .., x_{i+l}$ of $\mathbf{x}$ of size $l \leq L$ with each substring $x'_{j+1}, x'_{j+2}..., x'_{j+l}$ of $\mathbf{x}'$ having the same length. Each substring comparison yields a score that depends on the $\boldsymbol{\psi}$-similarity of their respective amino acids and a shifting contribution term that decays exponentially rapidly with the distance between the starting positions of the two substrings. The σ_p parameter controls the shifting contribution term. The σ_c parameter controls the amount of penalty incurred when the encoding vectors $\boldsymbol{\psi}^l(x_{i+1}, .., x_{i+l})$ and $\boldsymbol{\psi}^l(x'_{j+1}, .., x'_{j+l})$ differ as measured by the squared Euclidean distance between these two vectors. The GS kernel outputs the sum of all the substring-comparison scores.

Also, note that the GS kernel can be used on strings of different lengths, which is a great advantage over a localized string kernel (of fixed length) such as the RBF, the weighted degree kernels [Rätsch and Sonnenburg, 2004, Toussaint et al., 2010] or KISS [Jacob and Vert, 2008], a well known kernel method for the prediction of peptides binding to MHC-I. In fact, the GS kernel generalizes eight known kernels. Table 4.1 lists them with the fixed and free parameters. For example, when σ_p approaches $+\infty$ and σ_c approaches 0, the GS kernel becomes identical to the blended spectrum kernel [Shawe-Taylor and Cristianini, 2004], which has a free parameter L representing the maximum length of substrings. The free parameter values are usually determined by measuring the accuracy of the predictor on a separate ("hold-out") part of the data that was not used for training, or by more elaborate sampling methods such as cross-validation.

In contrast, Leslie et al. [2004] proposed the mismatch kernel which also extends the spectrum kernel, adding the important notion of mismatches (mutations) in the comparison of k-mers. This was motivated by the fact that mutations occur in proteins and thus k-mers should be considered up to a certain amount of mismatches. Not all mutations are equal, some will not affect the function of a protein as others will dramatically change the conformation of a protein or the binding affinity of a peptide. This is the motivating idea behind the $\boldsymbol{\psi}$ encoding function, amino acids properties are used to have a smooth transition between unimportant and critical mutations. Moreover, the transition can be adjusted thought the σ_c parameter.

Also, Saigo et al. [2004] proposed the local alignment (LA) kernel which sums all possible alignments with gaps between two sequences. The LA kernel is closely related to the popular

Fixed parameters	Free parameters	Kernel name
$L = 1, \sigma_p \to 0, \sigma_c \to 0$		Hamming distance
$L \to \infty, \sigma_p \to 0, \sigma_c \to 0$		Dirac delta
$\sigma_p \to \infty, \sigma_c \to 0$	L	Blended Spectrum [Shawe-Taylor and Cristianini, 2004]
$\sigma_p \to \infty$	L, σ_c	Blended Spectrum RBF [Toussaint et al., 2010]
$\sigma_c \to 0$	L, σ_p	Oligo [Meinicke et al., 2004]
$L \to \infty, \sigma_p \to 0$	σ_c	Radial Basis Function (RBF)
$\sigma_p \to 0, \sigma_c \to 0$	L	Weighted degree ($\star$) [Rätsch and Sonnenburg, 2004]
$\sigma_p \to 0$	L, σ_c	Weighted degree RBF ($\star$) [Toussaint et al., 2010]
	L, σ_p, σ_c	Generic String (GS)

TABLE 4.1 – Eight known kernels can be obtained by fixing different parameters of the GS kernel. ($\star$) Substituting $\boldsymbol{\psi}^l$ by $\boldsymbol{\psi}^l \sqrt{-\ln \beta_l}$ where the β_l's are the weighted degrees defined in Rätsch and Sonnenburg [2004].

Smith–Waterman alignment algorithm. In contrast, the GS kernel sums the contributions of all substrings according to their physicochemical properties with a position uncertainty penalising term. Also, the gap penalisation in the LA is well adapted to protein similarity by incorporating biological knowledge about protein evolution but not so much for identifying localized signals in sequences. Indeed, a small gap of only one amino acid in a peptide will have a dramatic influence on it's contacting residues and therefore on it's binding affinity. Finally, the LA kernel suffers from diagonal dominance, an issue the authors got around by taking the logarithm of the kernel. Unfortunately this operation does not preserve the positive definiteness of the kernel. However, the GS kernel does not suffer form diagonal dominance, thus avoiding many workarounds.

In the next subsection, we prove that the GS kernel is symmetric positive semi-definite and, therefore, defines a scalar product in some large-dimensional feature space (see Shawe-Taylor and Cristianini [2004]). In other words, for any hyperparameter values (L, σ_p, σ_c), there exists a function $\boldsymbol{\phi}_{\mathcal{X}_{(L,\sigma_p,\sigma_c)}}$ transforming each finite sequence of amino acids into a vector such that

$$GS(\mathbf{x}, \mathbf{x}', L, \sigma_p, \sigma_c) = \boldsymbol{\phi}_{\mathcal{X}_{(L,\sigma_p,\sigma_c)}}(\mathbf{x}) \cdot \boldsymbol{\phi}_{\mathcal{X}_{(L,\sigma_p,\sigma_c)}}(\mathbf{x}') \,.$$

Consequently, the solution minimizing the ridge regression functional $F(S, \mathbf{w})$ will be given by Equation (4.1) and is guaranteed to exist whenever the GS kernel is used.

Symmetric positive semi-definiteness of the GS kernel

The fact that the GS kernel is positive semi-definite follows from the following theorem. The proof is provided in Section 4.6.1.

Theorem 1. *Let Σ be an alphabet (say the alphabet of all the amino acids). For each $l \in \{1, .., L\}$, let $K_l : \Sigma^l \times \Sigma^l \longrightarrow \mathbb{R}$ be a symmetric positive semi-definite kernel. Let $A : \mathbb{R} \longrightarrow \mathbb{R}$ be any function which consists of a convolution of another function $B : \mathbb{R} \longrightarrow \mathbb{R}$ by itself. In other words, for all $z, z' \in \mathbb{R}$, we have*

$$A(z - z') \;\; = \;\; \int_{-\infty}^{+\infty} B(z - t)B(z' - t)\, \mathrm{d}t\,.$$

Then, the kernel K defined, for any two strings of length at least L on the alphabet Σ, as

$$K(\mathbf{x}, \mathbf{x}') \;\; \stackrel{def}{=} \;\; \sum_{l=1}^{L} \sum_{i=0}^{|\mathbf{x}|-l} \sum_{j=0}^{|\mathbf{x}'|-l} A(i-j) \;\; K_l\Big((x_{i+1}, .., x_{i+l})\,,\, (x'_{j+1}, .., x'_{j+l})\Big)$$

is also symmetric positive semi-definite.

The positive semi-definiteness of the GS kernel comes from the fact that the GS kernel is a particular case of the more general kernel K defined in the above theorem. Indeed, first note that both kernels are identical except $A(i-j)$ in kernel K is specialized to $\exp(\frac{-(i-j)^2}{2\sigma_p^2})$ in the GS kernel, and $K_l(\mathbf{y}\,,\,\mathbf{y}')$ in kernel K is specialized to $\exp(\frac{-\|\boldsymbol{\psi}^l(\mathbf{y}) - \boldsymbol{\psi}^l(\mathbf{y})\|^2}{2\sigma_c^2})$ in the GS kernel. Moreover, this last exponential is just an RBF kernel (see Shawe-Taylor and Cristianini [2004] for a definition) defined over vectors of $\mathbb{R}^{ld}$ of the form $\boldsymbol{\psi}^l(\mathbf{y})$; it is therefore positive semi-definite for any $l \in \{1, 2, .., L\}$. For the first exponential, note that $\exp(\frac{-(i-j)^2}{2\sigma_p^2})$ is a function that is obtained from a convolution of another function since we can verify that

$$\exp\left(\frac{-(i-j)^2}{2\sigma_p^2}\right) \;\; = \;\; \frac{\sqrt{2}}{\sigma_p\sqrt{\pi}} \int_{-\infty}^{+\infty} \exp\left(\frac{-(i-t)^2}{\sigma_p^2}\right) \exp\left(\frac{-(j-t)^2}{\sigma_p^2}\right) dt\,.$$

Indeed, this equality is a simple specialization of Equation (4.13) of Rasmussen and Williams [2006]. It is related to the fact that the convolution of two Normal distributions is still a Normal distribution.

Finally, it is interesting to point out that Theorem 1 can be generalized to any function A on measurable sets M (not only the ones that are defined on $\mathbb{R}$), provided that A is still is a convolution of another function $B : M \longrightarrow M$. We omit this generalized version in this paper since Theorem 1 suffices to prove that the GS kernel is positive semi-definite.

Efficient computation of the GS kernel

To cope with today's data deluge, the presented kernel should have a low computational cost. For this task, we first note that, before computing $GS(\mathbf{x}, \mathbf{x}', L, \sigma_p, \sigma_c)$ for each pair $(\mathbf{x}, \mathbf{x}')$ in the training set, we can first compute

$$E(a, a') \quad \stackrel{\text{def}}{=} \quad \|\boldsymbol{\psi}(a) - \boldsymbol{\psi}(a')\|^2 \quad = \quad \sum_{p=1}^{d} (\psi_p(a) - \psi_p(a'))^2,$$

for each pair (a, a') of amino acids. After this pre-computation stage, done in $O(d \cdot |\Sigma|^2)$ time, each access to $E(a, a')$ is done in $O(1)$ time. We will not consider the running time of this pre-computation stage in the complexity analysis of the GS kernel, because it only has to be done once to be used for any 5-tuple $(\mathbf{x}, \mathbf{x}', L, \sigma_p, \sigma_c)$. Recall that the binding affinity predictor, given by Equation 4.1, can be built only after we have computed the m^2 elements of the kernel matrix $\mathbf{K}$ (for a training set of m examples). Since m^2 is usually much larger than $d \cdot |\Sigma|^2$, we can omit this pre-computation time in the complexity analysis of kernel evaluations.

Now, recall that we have defined $\boldsymbol{\psi}^l : \Sigma^l \longrightarrow \mathbb{R}$ as the concatenation of vectors of the form $\boldsymbol{\psi}(a)$ (see Equation (4.2)). Hence, $\|\boldsymbol{\psi}^l(\mathbf{a}) - \boldsymbol{\psi}^l(\mathbf{a}')\|$ is an Euclidian norm, and we have

$$\|\boldsymbol{\psi}^l(\mathbf{a}) - \boldsymbol{\psi}^l(\mathbf{a}')\|^2 \quad = \quad \sum_{k=1}^{l} \|\boldsymbol{\psi}(a_k) - \boldsymbol{\psi}(a'_k)\|^2 \quad = \quad \sum_{k=1}^{l} E(a_k, a'_k) \tag{4.5}$$

Following this, we can now write the GS kernel as

$$GS(\mathbf{x}, \mathbf{x}', L, \sigma_p, \sigma_c) \quad = \quad \sum_{l=1}^{L} \sum_{i=0}^{|\mathbf{x}|-l} \sum_{j=0}^{|\mathbf{x}'|-l} e^{\left(\frac{-(i-j)^2}{2\sigma_p^2}\right)} \, e^{\left(\frac{-\sum_{k=1}^{l} E(x_{i+k}, x'_{j+k})}{2\sigma_c^2}\right)} \tag{4.6}$$

$$= \quad \sum_{i=0}^{|\mathbf{x}|} \sum_{j=0}^{|\mathbf{x}'|} e^{\left(\frac{-(i-j)^2}{2\sigma_p^2}\right)} \sum_{l=1}^{\min(L, |\mathbf{x}|-i, |\mathbf{x}'|-j)} e^{\left(\frac{-\sum_{k=1}^{l} E(x_{i+k}, x'_{j+k})}{2\sigma_c^2}\right)}, \tag{4.7}$$

where $\min(L, |\mathbf{x}| - i, |\mathbf{x}'| - j)$ is used in order to assure that $i + k$ and $j + k$ are valid positions in strings $\mathbf{x}$ and $\mathbf{x}'$.

Now, for any $L, |\mathbf{x}|, |\mathbf{x}'|$, and any $i \in \{1, \ldots, |\mathbf{x}|\}, j \in \{1, \ldots, |\mathbf{x}'|\}$, let

$$B_{i,j} \quad \stackrel{\text{def}}{=} \quad \sum_{l=1}^{\min(L, |\mathbf{x}|-i, |\mathbf{x}'|-j)} e^{\left(\frac{-\sum_{k=1}^{l} E(x_{i+k}, x'_{j+k})}{2\sigma_c^2}\right)}. \tag{4.8}$$

We therefore have

$$GS(\mathbf{x}, \mathbf{x}', L, \sigma_p, \sigma_c) = \sum_{i=0}^{|\mathbf{x}|} \sum_{j=0}^{|\mathbf{x}'|} \exp\left(\frac{-(i-j)^2}{2\sigma_p^2}\right) \cdot B_{i,j} \, . \tag{4.9}$$

Since $\min(L, |\mathbf{x}| - i, |\mathbf{x}'| - j) \leq L$, we see, from Equation (4.8), that the computation of each entry $B_{i,j}$ seems to involve $O(L^2)$ operations. However, we can reduce this complexity term to $O(L)$ by a dynamic programming approach. Indeed, consider the following recurrence :

$$t_k = \begin{cases} 1 & \text{if } k = 0 \\ t_{k-1} \cdot e^{\left(\frac{-E(x_{i+k}, x'_{j+k})}{2\sigma_c^2}\right)} & \text{otherwise.} \end{cases} \tag{4.10}$$

We thus have

$$B_{i,j} = \sum_{k=1}^{\min(L, |\mathbf{x}|-i, |\mathbf{x}'|-j)} t_k \tag{4.11}$$

The computation of each entry $B_{i,j}$ therefore involves only $O(L)$ operations. Consequently, the running time complexity of each GS kernel evaluation is $O\left(|\mathbf{x}| \cdot |\mathbf{x}'| \cdot L\right)$.

To test the efficiency of this dynamic programming algorithm, we conducted an experiment measuring the speedup obtained from using this algorithm versus a naïve implementation of Equation (4.4) that did not exploit dynamic programming. For peptides of length 15, 35 and 55, we measured the speedup obtained while computing 2500 kernel values as a function of the kernel parameter L.

For a given value of L, the speedup s is given by $s = t_n/t_d$, where t_n is the running time of the naïve implementation and t_d is the running time used by the dynamic programming algorithm.

The results shown in Figure 4.1 demonstrate that as the value of L increases, the dynamic programming algorithm is much more efficient than the naïve implementation.

GS Kernel approximation

In this section, we show how to compute a very close approximation of the GS kernel in linear time. Such a feature is interesting if one wishes to do a pre or post treatment where the symmetric positive semi-definite (SPSD) property of the kernel is not required. For example, as opposed to the training stage where the inverse of $\mathbf{K} + \mathbf{I}/C$ is guaranteed to exists only for a SPSD matrix $\mathbf{K}$, kernel values in the prediction stage could be approximated. Indeed, the scalar product with $\boldsymbol{\alpha}$ is defined for non positive semi-definite kernel values. This scheme would greatly speed up the predictions with a very small lost of accuracy and precision.

The shifting penalizing term, $\exp\left(\frac{-(i-j)^2}{2\sigma_p^2}\right)$ in Equation (4.4), implies that the further two substrings are from each other, no matter how similar they are, their contribution to the kernel will vanish exponentially rapidly. Let δ be the maximum distance between two substrings that we intend to consider in the computation of the approximate version of the GS kernel. In other words, any substring whose distance is greater than δ will not contribute. We propose to fix

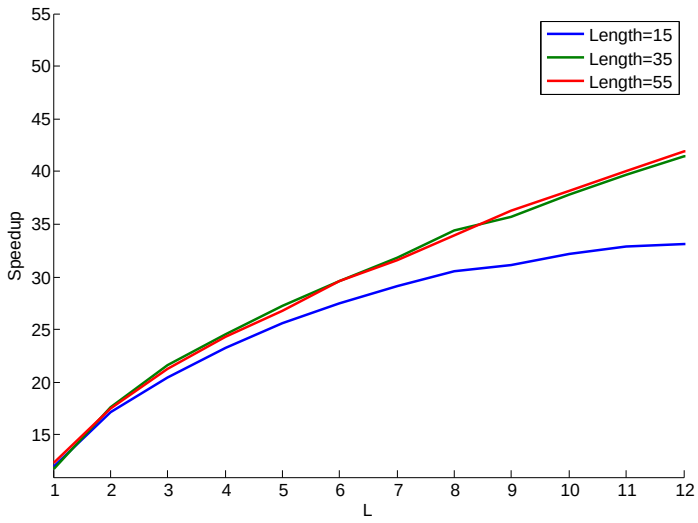


FIGURE 4.1 – This figure shows the speedup of the dynamic programming algorithm over a naïve implementation of the GS kernel as a function of the kernel parameter L. The running times were recorded while computing 2500 kernel values for peptides of length 15, 35 and 55. The other kernel parameters are $\sigma_p = 0.5$ and $\sigma_c = 0.5$.

$\delta = \lceil 3\sigma_p \rceil$. In this case, the contribution of any substring beyond δ is bound to be minimal. For the purpose of demonstration, let P be the $|\mathbf{x}| \times |\mathbf{x}'|$ matrix

$$P_{i,j} \overset{\text{def}}{=} \begin{cases} 0 & \text{if } |i-j| > \delta \\ \exp\left(\frac{-(i-j)^2}{2\sigma_p^2}\right) & \text{otherwise}\,. \end{cases} \tag{4.12}$$

P is thus a sparse matrix with exactly $\delta|\mathbf{x}| + \delta|\mathbf{x}'| - \delta^2$ non-zero values around it's diagonal. We can therefore write this approximation of the GS kernel as

$$GS'(\mathbf{x}, \mathbf{x}', L, \sigma_p, \sigma_c, \delta) = \sum_{i=0}^{|\mathbf{x}|} \sum_{j=0}^{|\mathbf{x}'|} P_{i,j} \cdot B_{i,j}\,. \tag{4.13}$$

It is clear that only values of B for which the value in P is non-zero need to be computed. The complexity of GS' is dominated by the computation of matrix B whose $\delta|\mathbf{x}| + \delta|\mathbf{x}'| - \delta^2$ entries can be computed in $O\left(\max(|\mathbf{x}|, |\mathbf{x}'|)\right)$. Since L and δ are constant factors, we have that $GS' \in O\left(\max(|\mathbf{x}|, |\mathbf{x}'|)\right)$, giving an optimal linear complexity.

To determine the speedup that can be obtained by approximating the GS kernel, we conducted an experiment measuring this speedup for different peptide lengths. For a given value of σ_p, the speedup s is given by $s = t_f/t_a$, where t_f is the time required for the computation using the GS kernel and t_a is the time required for the computation using the approximated GS kernel.

Figure 4.2 displays the speedups obtained for computing 1000000 kernel values with peptides of length 15, 35 and 55. We found that the approximation algorithm can greatly reduce the

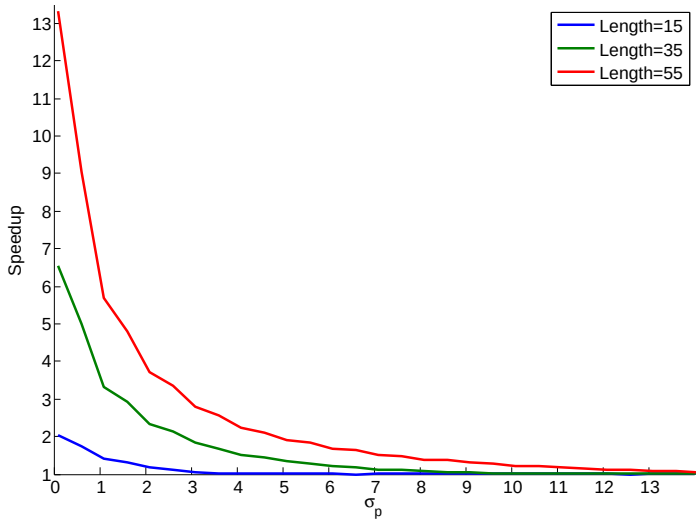


FIGURE 4.2 – This figure shows the speedup of the approximation algorithm over the full computation of the GS kernel as a function of the kernel parameter σ_p. The running times were recorded while computing 1000000 kernel values for peptides of length 15, 35 and 55. The other kernel parameters are $\sigma_c = 0.5$ and $L = 5$.

time required to compute kernel values. Note that, since the approximation algorithm only considers substrings of distance less than $\delta = \lceil 3\sigma_p \rceil$, for peptides of length l, the speedup obtained by using the approximation algorithm vanishes for $\sigma_p \geq l/3$.

4.2.3 Kernel for protein binding pocket

Hoffmann et al. [2010] proposed a new similarity measure between protein binding pockets. The similarity measure aligns atoms extracted from the binding pocket in $3D$ and assigns a score to the alignment. Pocket alignment is possible for proteins that share low sequence and structure similarity. They proposed two variations of the similarity measure. The first variation only compares the shape of pockets to assign a score. In the second variation, atom properties, such as partial charges, re-weight the contribution of each atom to the score. We will refer to these two variations respectively as sup-CK and sup-CK$_L$. Since both scores are invariant by rotation and translation, they are not positive semi-definite kernels. To obtain a valid kernel, we have used the so-called empirical kernel map where each $\mathbf{y}$ is mapped explicitly to $(k(\mathbf{y}_1, \mathbf{y}), k(\mathbf{y}_2, \mathbf{y}), \ldots, k(\mathbf{y}_m, \mathbf{y}))$. To ensure reproducibility and avoid implementation errors, all experiments were done using the implementation provided by the authors. An illustration of the pocket creation for the SupCk kernel is shown in Figure 4.3.

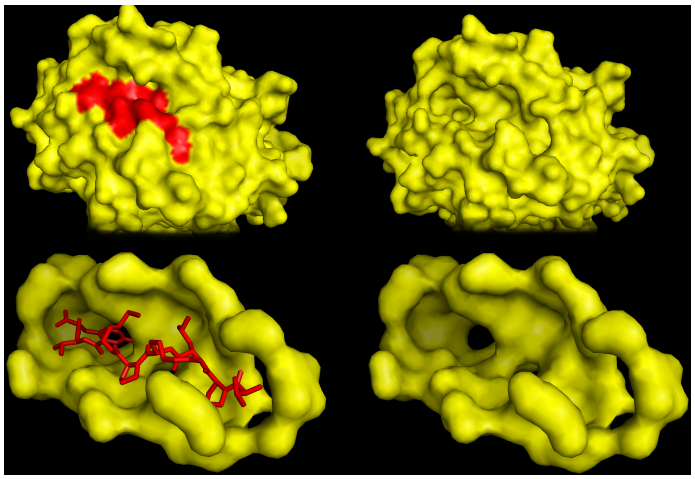


FIGURE 4.3 – This pyMOL illustration of a binding pocket, used for the binding pocket kernel[Hoffmann et al., 2010], shows a MHC-I molecule *B*3501* complexed with a peptide (VPLRPMTY) from the NEF protein of HIV1 (PDB ID 1A1N). The MHC protein is shown in yellow, the peptide is shown in red.

4.2.4 Kernel for protein structure

The MAMMOTH kernel is a similarity function between protein secondary structure proposed by Qiu et al. [2007]. This kernel is based on a sequence-independent structure alignment heuristic originally proposed by Ortiz et al. [2002]. Structural information from crystals is used to align two proteins using their secondary structure, a score is assigned to the alignment. The greater the similarity between the two proteins' secondary structure, the greater the alignment score will be. Ortiz et al. [2002] showed that the heuristic was able to produce an accurate alignment for both high and low resolution structures. Also, this kernel was recently used with success for prediction of protein-protein interactions [Hue et al., 2010]. To ensure reproducibility and avoid implementation errors, all experiments were done using the implementation provided by the authors.

4.2.5 Metrics and experimental design

When dealing with regression values, classical metrics used for classification such as the area under the ROC curve (AUC) [Swets, 1988] are not suitable. To compute the AUC, some authors determine a binding affinity threshold value and use it to transform the regression problem into a binary classification problem. The real value outputs of the predictor are then mapped to binary classes using the threshold and the AUC is computed using these binary values. Unfortunately, this approach makes the value of the AUC metric dependent on the chosen treshold value. For this reason, we decided not to present results for the AUC metric in this paper. Nevertheless, these results are provided in Section 4.6.2.

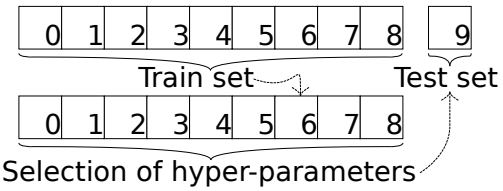


FIGURE 4.4 – Nested 10-fold cross-validation. For each of the 10 outer folds, an inner 9 fold cross-validation scheme was used to select hyperparameters.

Fortunately, metrics such as the root mean squared error (RMSE), the coefficient of determination (R^2), and the Pearson product-moment correlation coefficient (PCC) are more suited for measuring the performance of predictors on regression problems. Therefore, in this paper, we have used the PCC and the RMSE to evaluate the performance of our method.

Except when otherwise stated, 10 folds nested cross-validation was done for estimating the PCC and the RMSE of the predicted binding affinities (See Figure 4.4). For all n (here $n = 10$) outer folds, $n - 1$ inner cross-validation folds were used for the selection of the kernel hyperparameters and the C parameter of Equation (4.1). Note that, all reported values were computed on the union of the outer fold test set predictions. This is important, since an average of correlation coefficients is not a valid correlation coefficient. This is also true for the root mean squared error.

More precisely, let $\bar{e}$ denote the average affinity in the data set $\mathcal{D}$. Let T_k for $k \in \{1, \ldots, 10\}$ denote the testing set of the k^{th} outer fold and let $h_{\mathcal{D}\setminus T_k}(\mathbf{x}_i, \mathbf{y}_i)$ be the predicted binding affinity on example $((\mathbf{x}_i, \mathbf{y}_i), e_i)$ of the predictor built from $\mathcal{D} \setminus T_k$. The correlation coefficient was computed using :

$$PCC = \sqrt{1 - \frac{\sum_{k=1}^{n} \sum_{i \in T_k} \left(e_i - h_{\mathcal{D}\setminus T_k}(\mathbf{x}_i, \mathbf{y}_i)\right)^2}{\sum_{i \in \mathcal{D}} \left(e_i - \bar{e}\right)^2}}\,. \tag{4.14}$$

An algorithm that, on average, produces a predictor that makes the same quadratic error as the constant predictor $\bar{e}$ will give $PCC = 0$ and an algorithm that always returns a perfect predictor will give $PCC = 1$.

As for the RMSE, it was computed using

$$RMSE = \sqrt{\frac{\sum_{k=1}^{n} \sum_{i \in T_k} (e_i - h_{\mathcal{D}\setminus T_k}(\mathbf{x}_i, \mathbf{y}_i))^2}{|\mathcal{D}|}} \tag{4.15}$$

Therefore, the perfect predictor will give $RMSE = 0$ and the value of this metric will increase as the quality of the predictor decrease.

All the p-values reported in this article were computed using the two-tailed Wilcoxon signed-ranked test.

Finally, for all the experiments, hyperparameters for the GS kernels and the learning algorithms were selected by grid search using the following ranges : $C \in [0, 100]$, $\sigma_p \in]0, 18]$, $\sigma_c \in]0, 18]$ and $L \in [1, 15]$.

4.2.6 Data

PepX database

The PepX database [Vanhee et al., 2010] contains 1431 high-quality peptide-protein complexes along with their protein and peptide sequences, high quality crystal structures, and binding energies (expressed in kcal/mol) computed using the FoldX force field method. Full diversity of structural information on protein-peptide complexes is achieved with peptides bound to, among others, MHC, thrombins, α-ligand binding domains, SH3 domains and PDZ domains. This database recently drew attention in a review on the computational design of peptide ligands [Vanhee et al., 2011] where it was part of large structural studies to understand the specifics of peptide binding. A subset of 505 non-redundant complexes was selected based on the dissimilarity of their binding interfaces. The authors of the database performed the selection in such a way that this smaller subset still represented the full diversity of structural information on peptide-protein complexes present in the entire Protein Data Bank (PDB), see Vanhee et al. [2010] for a description of the method. We will refer to the smaller subset as the "PepX Unique" data set and to the whole data base as "PepX All".

The few complexes with positive binding energies were removed from the dataset. No other modifications were made to the original database.

Major histocompatibility complex class II (MHC-II)

Two different approaches were used for the prediction of MHC class II peptide binding affinities : single-target and multi-target (pan-specific).

Single-target prediction experiments were conducted using the data from the IEDB dataset proposed by the authors of the RTA method [Bordner and Mittelmann, 2010b]. The latter consists of 16 separate datasets, each containing data on the peptides binding to an MHC class II allotype. For each allotype, the corresponding dataset contains the binding peptide sequences and their binding affinity in kcal/mol. These datasets have previously been separated into 5 cross-validation folds to minimize overlapping between peptide sequences in each fold. It is well known in the machine learning community that such practice should be avoided, as opposed to random fold selection, since the training and test sets should be independently generated. These predefined folds were nevertheless used for the purpose of comparison with other learning methodologies that have used them.

Pan-specific experiments were conducted on the IEDB dataset proposed by the authors of the NetMHCIIpan method [Nielsen et al., 2008]. The dataset contains 14 different HLA-DR allotypes, with 483 to 5648 binding peptides per allotype. For each complex, the dataset contains the HLA allele's identifier (e.g. : *DRB1*0101*), the peptide's sequence and the $\log 50k$ transformed IC50 (Inhibitory Concentration 50%), which is given by $1 - \log_{50000} IC50$.

As pan-specific learning requires comparing HLA alleles using a kernel, the allele identifiers contained in the dataset were not directly usable for this purpose. Hence, to obtain a useful similarity measure (or kernel) for pairs of HLA alleles, we used the pseudo sequences composed of the amino acids at highly polymorphic positions in the alleles' sequences. These amino acids are potentially in contact with peptide binders[Nielsen et al., 2008], therefore contributing to the MHC molecule's binding specificity. The authors of the NetMHCIIpan method proposed using pseudo sequences composed of the amino acids at 21 positions that were observed to be polymorphic for HLA-DR, DP and DQ [Nielsen et al., 2008]. With respect to the IMGT nomenclature [Robinson et al., 2000], these amino acids are located between positions 1 and 89 of the MHC's β chain. Pseudo sequences consisting of all 89 amino acids between these positions were also used to conduct the experiments.

Quantitative structure affinity model (QSAM) benchmark

Three well-studied benchmark datasets for designing quantitative structure affinity models were also used to compare our approach : 58 angiotensin-I converting enzyme (ACE) inhibitory dipeptides, 31 bradykinin-potentiating pentapeptides and 101 cationic antimicrobial pentadecapeptides. These data sets were recently the subject of extensive studies [Zhou et al., 2010] where partial least squares (PLS), Artificial Neural Networks (ANN), Support Vector Regression (SVR), and Gaussian Processes (GP) were used to predict the biological activity of the peptides. GP and SVR were found to have the best results on the testing set, but their experiment protocol was unconventional because the training and test sets were not randomly selected from the data set. Instead, their testing examples were selected from a cluster analysis performed on the whole data set—thus favoring learning algorithms that tend to cluster their predictions according to the same criteria used to split the data. Instead, we randomly selected the testing examples from the whole data set—thus avoiding a bias that would favor some algorithms *a priori*. Theses datasets were chosen to demonstrate the ability of our method to learn on both small and large datasets.

4.3 Results and discussion

4.3.1 PepX database

To our knowledge, this is the first kernel method attempt at learning a predictor which takes the protein crystal and the peptide sequence as input to predict the binding energy of the

	SVR	KRR					
	sup-CK	sup-CK		BS	MAMMOTH	sup-CK$_L$	
	BS	BS	GS	BS	BS	BS	GS
PepX Unique	0.6822	0.7072	**0.7300**	0.5873	0.5828	0.7110	0.7264
PepX All	0.8227	0.8580	0.8648	0.7769	0.8152	0.8601	**0.8652**

TABLE 4.2 – Correlation coefficient (PCC) for multiple target predictions on the PepX database. Best results are highlighted in bold.

complex. Many consider this task as a major challenge with important consequences for molecular biology. Standard string kernels for protein primary structures such as the LA-kernel and the blended spectrum (BS) were used while conducting experiments on proteins. They did not yield good results, mainly because they do not consider the protein's secondary structure information. To validate this hypothesis and improve our results, we tried using the MAMMOTH kernel. The MAMMOTH kernel did improve the results (see Table 4.2) over the blended spectrum (BS) but was still missing an important aspect of protein-peptide interaction. The interaction takes place at a very specific location on the surface of the protein called the binding pocket. Two proteins may be very different, but if they share a common binding pocket, it is likely that they will bind similar ligands. This is the core idea that motivated the design of the sup-CK binding pocket kernel [Hoffmann et al., 2010].

Choosing a kernel for the peptides was also a challenging task. Sophisticated kernels for local signals such as the RBF, the weighted degree, and the weighted degree RBF could not be used because peptide lengths were not equal. In fact, peptide lengths vary between 5 and 35 amino acids, which makes the task of learning a predictor and designing a kernel even more challenging. This was part of our motivation in designing the GS kernel. For all experiments, the BLOSUM 50 matrix was found to be the optimal amino acid descriptors during cross-validation.

Table 4.2 presents the first machine learning results for the prediction of binding affinity given any peptide-protein pair. We first observe that KRR has better accuracy than SVR. We also note that using the GS kernel over the simpler BS kernel improves the accuracy for both the sup-CK and the sup-CK$_L$ kernels for binding pockets. It is surprising that the sup-CK$_L$ kernel does not outperform the sup-CK kernel on both benchmarks, since the addition of the atom partial charges should provide more relevant information to the predictor.

Figures 4.5 and 4.6 present an illustration of the prediction accuracy using sup-CK for the PepX Unique dataset and sup-CK$_L$ for the PepX All dataset. For illustration purposes, the absolute value of the binding energy has been plotted. We observe that the predictor has the property of maintaining ranking of binding affinities. Consequently, peptides with high binding

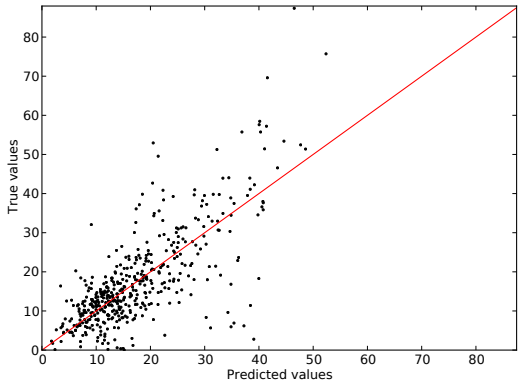


FIGURE 4.5 – Predicted values for all peptide-protein complexes as a function of the true value for the PepX Unique dataset. A perfect predictor would have all it's predictions lying on the $y = x$ red line.

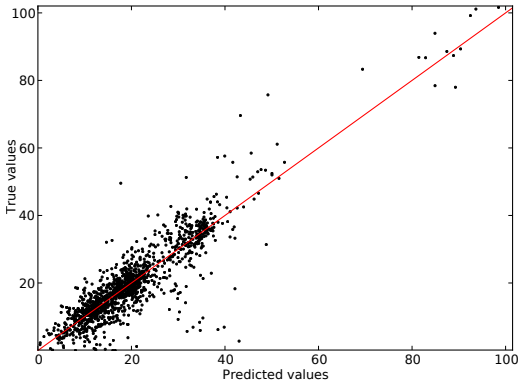


FIGURE 4.6 – Predicted values for all peptide-protein complexes as a function of the true value for the PepX All dataset. A perfect predictor would have all it's predictions lying on the $y = x$ red line.

affinity can generally be identified—an important feature for drug discovery. Peptides with the highest binding affinities are the ones that, ultimately, will serve as precursor drug or scaffold in a rational drug design program.

Experiments showed that a Pearson correlation coefficient of ≈ 1.0 is attainable on the training set when using the binding pocket kernel, the GS kernel and a large value for the complexity-accuracy trade-off parameter C (empirically ≈ 100), thus giving little weight to the regularization term. This is a strong indication that the proposed method has the ability of building a good predictor, but the lack of data quality and quantity may be responsible for the reduced performance on the testing set. Hence better data may improve the quality of the predictor. Initially, biological validation will be necessary but ultimately, when sufficient data is gathered, the predictor may provide accurate results that are currently only achievable by high cost

biological experimentation.

4.3.2 Major histocompatibility complex class II (MHC-II)

Single-target predictions

We performed a single-target prediction experiment using the dataset proposed by the authors of the RTA method [Bordner and Mittelmann, 2010b]. The goal of such experiments was to evaluate the ability of a predictor to predict the binding energy (kcal/mol) of an unknown peptide to a specific MHC allotype when training only on peptides binding to this allotype. For each of the 16 MHC allotypes, a predictor was trained using kernel ridge regression with the GS kernel and a nested cross-validation scheme was used. For comparison purposes, the nested cross-validation was done using the 5 predefined cross-validation folds provided in Bordner and Mittelmann [2010b]. Again, this is sub-optimal from the statistical machine learning perspective, since the known guarantees on the risk of a predictor [Shawe-Taylor and Cristianini, 2004, Schölkopf and Smola, 2002] normally require that the examples be generated independently from the same distribution.

Three common metrics were used to compare the methods : the Pearson correlation coefficient (PCC), the root mean squared error (RMSE), and the area under the ROC curve (AUC). The PCC and the RMSE results are presented in Table 4.3, AUC values can be found as supplementary material [see Additional File 2]. The PCC results show that our method significantly outperforms the RTA method on 13 out of 16 allotypes with a p-value of 0.0308. The inferior results for certain allotypes may be attributed to the small size of these datasets. In addition, the RMSE results show that our method clearly outperforms the RTA method on all 16 allotypes with a p-value of 0.0005.

Pan-specific predictions

To evaluate the performance of our method and the potential of the GS kernel, pan-specific predictions were performed using the dataset proposed by the authors of NetMHCIIpan [Nielsen et al., 2008]. The authors proposed a new cross-validation scheme called the *leave one allele out* (LOAO) where all but one allele are used as training set and the remaining allele is used as testing set. This is a more challenging problem, as the predictor needs to determine the binding affinity of peptides for an allele which was absent in the training data. The binding specificity of an allele's interface is commonly characterized using a pseudo sequence extracted from the beta chain's sequence [Zhang et al., 2012, Nielsen et al., 2008, 2010]. During our experiments, the 21 amino acid pseudo sequences were found to be optimal. The 89 amino acid pseudo sequences yielded similar, but slightly suboptimal results. For all experiments, the GS kernel was used for the allele pseudo sequences and for the peptide sequences. All results were obtained with the same LOAO scheme presented in Nielsen et al. [2008]. For each allele, an inner LOAO cross-validation was done for the selection of hyperparameters.

MHC β chain	PCC		RMSE (kcal/mol)		# of examples
	KRR+GS	RTA	KRR+GS	RTA	
DRB1*0101	**0.632**	0.530	**1.20**	1.43	5648
DRB1*0301	**0.538**	0.425	**1.16**	1.46	837
DRB1*0401	**0.430**	0.340	**1.44**	1.72	1014
DRB1*0404	**0.491**	0.487	**1.25**	1.38	617
DRB1*0405	**0.530**	0.442	**1.09**	1.35	642
DRB1*0701	**0.645**	0.484	**1.24**	1.62	833
DRB1*0802	**0.469**	0.412	**1.19**	1.34	557
DRB1*0901	0.303	**0.369**	**1.55**	1.68	551
DRB1*1101	**0.550**	0.450	**1.17**	1.45	812
DRB1*1302	**0.468**	0.464	**1.51**	1.64	636
DRB1*1501	**0.502**	0.438	**1.41**	1.53	879
DRB3*0101	0.380	**0.425**	**1.03**	1.13	483
DRB4*0101	**0.613**	0.522	**1.10**	1.33	664
DRB5*0101	**0.541**	0.434	**1.20**	1.57	835
H2*IA$_b$	**0.603**	0.556	**1.00**	1.15	526
H2*IA$_d$	0.325	**0.563**	**1.44**	1.53	306
Average :	**0.501**	0.459	**1.25**	1.46	

TABLE 4.3 – Comparison of HLA-DR prediction results on the dataset proposed by the authors of RTA. Best results for each metric are highlighted in bold. The PCC results show that the proposed method (KRR+GS) outperforms the RTA method with a p-value of 0.0308. The RMSE results show that KRR+GS outperforms the RTA method on all 16 allotypes with a p-value of 0.0005.

To assess the performance of the proposed method, the PCC and the RMSE results are shown in Table 4.4, AUC values can be found in the supplementary material [see Additional File 2]. Since we performed LOAO cross-validation, the PCC, RMSE and AUC values were calculated on each test fold individually, thus yielding results for each allele.

The PCC results show that our method outperforms the MultiRTA[Bordner and Mittelmann, 2010a] (p-value of 0.001) and the NetMHCIIpan-2.0[Nielsen et al., 2010] (p-value of 0.0574) methods. Since the dataset contained values in $\log 50k$ transformed IC50 (Inhibitory Concentration 50%), the calculation of the RMSE values required converting the predicted values to kcal/mol using the method proposed in Bordner and Mittelmann [2010b].

The RMSE values are only shown for our method and the MultiRTA method, since such values were not provided by the authors of NetMHCIIpan-2.0. The RMSE results indicate that our method globally outperforms MultiRTA with a p-value of 0.0466.

MHC β chain	PCC			RMSE (kcal/mol)		# of examples
	KRR+GS	MultiRTA	NetMHCIIpan-2.0	KRR+GS	MultiRTA	
DRB1*0101	**0.662**	0.619	0.627	1.48	**1.33**	5166
DRB1*0301	**0.743**	0.438	0.560	**1.29**	1.36	1020
DRB1*0401	**0.667**	0.534	0.652	**1.36**	1.56	1024
DRB1*0404	0.709	0.623	**0.731**	**1.18**	1.33	663
DRB1*0405	0.606	0.566	**0.626**	**1.25**	1.28	630
DRB1*0701	0.694	0.620	**0.753**	**1.34**	1.51	853
DRB1*0802	**0.728**	0.523	0.700	**1.23**	1.45	420
DRB1*0901	0.471	0.375	**0.474**	**1.53**	2.01	530
DRB1*1101	**0.786**	0.603	0.721	**1.16**	1.46	950
DRB1*1302	**0.416**	0.365	0.337	1.73	**1.68**	498
DRB1*1501	**0.612**	0.513	0.598	**1.46**	1.57	934
DRB3*0101	**0.654**	0.603	0.474	1.52	**1.10**	549
DRB4*0101	**0.540**	0.508	0.515	**1.41**	1.61	446
DRB5*0101	**0.732**	0.543	0.722	**1.28**	1.60	924
Average :	**0.644**	0.531	0.606	**1.37**	1.49	

TABLE 4.4 – Comparison of pan-specific HLA-DR prediction results on the dataset proposed by the authors of NetMHCIIpan. Best results for each metric are highlighted in bold. The PCC results show that the proposed method (KRR+GS) outperforms MultiRTA with a p-value of 0.001 and NetMHCIIpan-2.0 with a p-value of 0.0574. The RMSE results indicate that KRR+GS outperforms MultiRTA with a p-value of 0.0466.

	SVR	KRR	
	RBF	RBF	GS
ACE	0.8782	0.8807	**0.9044**
Bradykinin	0.7491	0.7531	**0.7641**
Cationic	0.7511	0.7417	**0.7547**

TABLE 4.5 – Correlation coefficient (PCC) on the QSAM benchmarks. Best results are highlighted in bold.

4.3.3 Quantitative structure affinity model (QSAM) benchmark

For all datasets, the extended z scale [Zhou et al., 2010] was found to be the optimal amino acids descriptors during cross-validation. All the results in this section were thus obtained using the extended z scale for the RBF and GS kernels. All peptides within each data set are of the same length, which is why the RBF kernel can be applied, as opposed to the PepX database or the two MHC-II benchmark datasets. Note the RBF kernel is a special case of the GS kernel. Hence, the results obtained from our method using the GS kernel were likely to be at least as good as those obtained with the RBF kernel.

Table 4.5 present the results obtained when applying the method from Zhou et al. [2010] (SVR learning with the RBF kernel) and our method (KRR learning with the GS kernel). Results with the RBF kernel and KRR are also presented to illustrate the gain in accuracy obtained

from the more general GS kernel.

We observed that kernel ridge regression (KRR) had a slight accuracy advantage over support vector regression (SVR). Moreover, SVR has one more hyperparameter to tune than KRR : the ϵ-insensitive parameter. Consequently, KRR should be preferred over SVR for requiring a substantially shorter learning time. Also, we show in Table 4.5 that the GS kernel outperforms the RBF kernel on all three QSAM data sets (when limiting ourself to KRR). Considering these results, KRR with the GS kernel clearly outperforms the method of Zhou et al. [2010] on all data sets.

4.3.4 Additionnal results and external validation

To act as an external source of validation for our results and to assess the performance of the GS kernel, we participated in the 2012 Machine Learning Competition in Immunology [Dana-Farber Cancer Institute, 2012]. The goal of this competition was to identify, given unpublished experimental data, which new peptides were naturally processed by MHC Class I pathway for 8 target molecules. Our method achieved the best prediction performance for HLA-B*0702, HLA-B*5301, H2-Db, and H2-Kb molecules, validating the suitability of the GS kernel for such problems.

These results support our claim that the GS kernel is a state-of-the-art kernel for peptides and a valuable tool for computationnal biologists.

4.4 Conclusions

We have proposed a new kernel designed for small bio-molecules (such as peptides) and pseudo-sequences of binding interfaces. The GS kernel is an elegant generalization of eight known kernels for local signals. Despite the richness of this new kernel, we have provided a simple and efficient dynamic programming algorithm for its exact computation and a linear time algorithm for its approximation. Combined with the kernel ridge regression learning algorithm and the binding pocket kernel, the proposed kernel yields promising results on the PepX database. For the first time, a predictor capable of accurately predicting the binding affinity of any peptide to any protein was learned using this database. Our method significantly outperformed RTA on the single-target prediction of MHC-II binding peptides. Impressive state-of-the-art results were also obtained on the pan-specific MHC-II task, outperforming both MultiRTA and NetMHCIIpan-2.0. Moreover, the method was successfully tested on three well studied datasets for the quantitative structure affinity model.

A predictor trained on the whole IEDB database or PDB database, as opposed to benchmark datasets, would be a substantial tool for the community. Unfortunately, learning a predictor on very large datasets (over 25000 examples) is still a major challenge with most machine

learning methods, as the similarity (Gram) matrix becomes hard to fit into the memory of most computers. We propose to expand the presented method to very large datasets as future work. The method, the kernel and benchmark datasets will be made available online after publication.

4.5 Acknowledgements

Computations were performed on the SciNet supercomputer at the University of Toronto, under the auspice of Compute Canada. The operations of SciNet are funded by the Canada Foundation for Innovation (CFI), the Natural Sciences and Engineering Research Council of Canada (NSERC), the Government of Ontario and the University of Toronto. JC is the Canada Research Chair in Medical Genomics. This work was supported in part by the Fonds de recherche du Québec - Nature et technologies (FL, MM & JC ; 2013-PR-166708) and the NSERC Discovery Grants (FL ; 262067, MM ; 122405).

4.6 Appendix

4.6.1 The proof of theorem 1

Theorem 1. *Let Σ be an alphabet (say the alphabet of all the amino acids). For each $l \in \{1, .., L\}$, let $K_l : \Sigma^l \times \Sigma^l \longrightarrow \mathbb{R}$ be a symmetric positive semi-definite kernel. Let $A : \mathbb{R} \longrightarrow \mathbb{R}$ be any function which consists of a convolution of another function $B : \mathbb{R} \longrightarrow \mathbb{R}$ by itself. In other words, for all $z, z' \in \mathbb{R}$, we have*

$$A(z - z') \quad = \quad \int_{-\infty}^{+\infty} B(z - t)B(z' - t)\,\mathrm{d}t\,.$$

Then, the kernel K defined, for any two strings of length at least L on the alphabet Σ, as

$$K(\mathbf{x}, \mathbf{x}') \quad \stackrel{def}{=} \quad \sum_{l=1}^{L} \sum_{i=0}^{|\mathbf{x}|-l} \sum_{j=0}^{|\mathbf{x}'|-l} A(i-j) \;\; K_l\Big((x_{i+1}, .., x_{i+l})\,,\,(x'_{j+1}, .., x'_{j+l})\Big)$$

is also symmetric positive semi-definite.

Let us first recall the well-known definition (due to Mercer (1909)) of a PSD kernel that we state here in its "string kernel" version. See also Cristianini and Shawe-Taylor [2000] for a similar formulation.

Definition 2. [Mercer] *Let Σ be a finite alphabet and $\mathcal{D} \subseteq \Sigma^\star$, a set of words on that alphabet. Then, a symmetric string kernel $\tilde{K} : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}$ is* positive semi-definite *if and only if it satisfies Mercer's condition,* i.e.*,*

$$\sum_{\mathbf{x} \in \mathcal{D}} \sum_{\mathbf{x}' \in \mathcal{D}} f(\mathbf{x}) f(\mathbf{x}') \tilde{K}(\mathbf{x}, \mathbf{x}') \quad \geq \quad 0$$

for any function $f : \mathcal{D} \longrightarrow \mathbb{R}$ *such that* $\sum_{\mathbf{x}\in\mathcal{D}} f^2(\mathbf{x}) < +\infty$.

Proof of Theorem 1 : For compactness of the notation, the sub-string $(x_{i+1}, x_{i+2}, \ldots, x_{i+l})$ of a string $\mathbf{x}$ is denoted $\mathbf{x}_{]i:i+l]}$. Also, the set of all strings of of length at least (respectively at most) k on the alphabet Σ is denoted $\Sigma^{\geq k}$ (respectively $\Sigma^{\leq k}$). Clearly, the kernel K is symmetric and has domain $\Sigma^{\geq L} \times \Sigma^{\geq L}$. So, we only have to show that it satisfies Mercer's condition of Definition 2 (with $\mathcal{D} = \Sigma^{\geq L}$ and $\tilde{K} = K$). Consequently, we have

$$
\begin{aligned}
&\sum_{\mathbf{x}\in\Sigma^{\geq L}} \sum_{\mathbf{x}'\in\Sigma^{\geq L}} f(\mathbf{x}) f(\mathbf{x}') K(\mathbf{x}, \mathbf{x}') \\
&\quad = \sum_{s=L}^{+\infty} \sum_{s'=L}^{+\infty} \sum_{\mathbf{x}\in\Sigma^s} \sum_{\mathbf{x}'\in\Sigma^{s'}} f(\mathbf{x}) f(\mathbf{x}') K(\mathbf{x}, \mathbf{x}') \qquad (4.16) \\
&\quad = \sum_{s=L}^{+\infty} \sum_{s'=L}^{+\infty} \sum_{\mathbf{x}\in\Sigma^s} \sum_{\mathbf{x}'\in\Sigma^{s'}} \sum_{l=1}^{L} \sum_{i=0}^{|\mathbf{x}|-l} \sum_{j=0}^{|\mathbf{x}'|-l} A(i-j)\ f(\mathbf{x}) f(\mathbf{x}') K_l(\mathbf{x}_{]i,i+l]}, \mathbf{x}'_{]j,j+l]}) \\
&\quad = \sum_{s=L}^{+\infty} \sum_{s'=L}^{+\infty} \sum_{\mathbf{x}\in\Sigma^s} \sum_{\mathbf{x}'\in\Sigma^{s'}} \sum_{l=1}^{L} \sum_{i=0}^{s-l} \sum_{j=0}^{s'-l} A(i-j)\ f(\mathbf{x}) f(\mathbf{x}') K_l(\mathbf{x}_{]i,i+l]}, \mathbf{x}'_{]j,j+l]}) \\
&\quad = \sum_{l=1}^{L} \sum_{s=L}^{+\infty} \sum_{s'=L}^{+\infty} \sum_{i=0}^{s-l} \sum_{j=0}^{s'-l} A(i-j) \sum_{\mathbf{x}\in\Sigma^s} \sum_{\mathbf{x}'\in\Sigma^{s'}} f(\mathbf{x}) f(\mathbf{x}') K_l(\mathbf{x}_{]i,i+l]}, \mathbf{x}'_{]j,j+l]}) \qquad (\star)
\end{aligned}
$$

Note that the last part of the last line of Equation $(\star)$ can be rewritten as

$$
\begin{aligned}
&A(i-j) \sum_{\mathbf{x}\in\Sigma^s} \sum_{\mathbf{x}'\in\Sigma^{s'}} f(\mathbf{x}) f(\mathbf{x}') K_l(\mathbf{x}_{]i,i+l]}, \mathbf{x}'_{]j,j+l]}) \\
&\quad = A(i-j) \sum_{\mathbf{y}\in\Sigma^l} \sum_{\mathbf{y}'\in\Sigma^l} \sum_{\mathbf{x}\in\Sigma^s | \mathbf{x}_{]i:i+l]}=\mathbf{y}} \sum_{\mathbf{x}'\in\Sigma^{s'} | \mathbf{x}'_{]i:i+l]}=\mathbf{y}'} f(\mathbf{x}) f(\mathbf{x}') K_l(\mathbf{y}, \mathbf{y}') \\
&\quad = \int_{-\infty}^{+\infty} B(i-t) B(j-t)\, \mathrm{d}t \sum_{\mathbf{y}\in\Sigma^l} \sum_{\mathbf{y}'\in\Sigma^l} \sum_{\mathbf{x}\in\Sigma^s | \mathbf{x}_{]i:i+l]}=\mathbf{y}} \sum_{\mathbf{x}'\in\Sigma^{s'} | \mathbf{x}'_{]i:i+l]}=\mathbf{y}'} f(\mathbf{x}) f(\mathbf{x}') K_l(\mathbf{y}, \mathbf{y}')\,, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\star\star)
\end{aligned}
$$

where $\mathbf{x} \in \Sigma^s \mid \mathbf{x}_{]i:i+l]} = \mathbf{y}$ means that the strings $\mathbf{x}$ is any string of length s that contains the substring $\mathbf{y}$ starting at position $i+1$.

Now, for any integers $s \geq L$ and $l \geq 1$, let us define $g_{s,l} : \Sigma^l \times \mathbb{R} \longrightarrow \mathbb{R}$, as

$$
g_{s,l}(\mathbf{y}, t) \stackrel{\text{def}}{=} \sum_{i=0}^{s-l} B(i-t) \sum_{\mathbf{x}\in\Sigma^s | \mathbf{x}_{]i:i+l]}=\mathbf{y}} f(\mathbf{x})\,. \qquad (4.17)
$$

Note that $g_{s,l}(\mathbf{y}, t)$ is finite because it is a double sum, each of which being finite. It therefore follows follows that

$$\sum_{\mathbf{y}\in\Sigma^l} [g_{s,l}(\mathbf{y}, t)]^2 \;<\; \infty\,, \qquad (\star\star\star)$$

because this is also a finite sum. Now, combining $(\star)$ and $(\star\star)$, we obtain

$$\begin{aligned}
&\sum_{\mathbf{x}\in\Sigma^{\geq L}} \sum_{\mathbf{x}'\in\Sigma^{\geq L}} f(\mathbf{x}) f(\mathbf{x}') K(\mathbf{x}, \mathbf{x}') \\
&\quad = \sum_{l=1}^{L} \sum_{(\mathbf{y},\mathbf{y}')\in\Sigma^l\times\Sigma^l} K_l(\mathbf{y}, \mathbf{y}') \sum_{s=L}^{+\infty} \sum_{i=0}^{s-l} \sum_{s'=L}^{+\infty} \sum_{j=0}^{s'-l} \int_{-\infty}^{+\infty} B(i-t) B(j-t)\, \mathrm{d}t \\
&\qquad \times \left[\sum_{\mathbf{x}\in\Sigma^s | \mathbf{x}_{]i:i+l]} = \mathbf{y}} f(\mathbf{x}) \right] \left[\sum_{\mathbf{x}'\in\Sigma^{s'} | \mathbf{x}'_{]j:j+l]} = \mathbf{y}'} f(\mathbf{x}') \right] \\
&\quad = \sum_{l=1}^{L} \sum_{s=L}^{+\infty} \sum_{s'=L}^{+\infty} \int_{-\infty}^{+\infty} \left(\sum_{(\mathbf{y},\mathbf{y}')\in\Sigma^l\times\Sigma^l} K_l(\mathbf{y}, \mathbf{y}') \left[\sum_{i=0}^{s-l} B(i-t) \sum_{\mathbf{x}\in\Sigma^s | \mathbf{x}_{]i:i+l]} = \mathbf{y}} f(\mathbf{x}) \right] \right. \\
&\qquad \left. \times \left[\sum_{j=0}^{s'-l} B(j-t) \sum_{\mathbf{x}'\in\Sigma^{s'} | \mathbf{x}'_{]j:j+l]} = \mathbf{y}'} f(\mathbf{x}') \right] \right) \mathrm{d}t \\
&\quad = \sum_{l=1}^{L} \sum_{s=L}^{+\infty} \sum_{s'=L}^{+\infty} \int_{-\infty}^{+\infty} \left(\sum_{(\mathbf{y},\mathbf{y}')\in\Sigma^l\times\Sigma^l} K_l(\mathbf{y}, \mathbf{y}') g_{s,l}(\mathbf{y}, t) g_{s',l}(\mathbf{y}', t) \right) \mathrm{d}t \\
&\qquad \geq 0\,.
\end{aligned}$$

The last line follows from Definition 2 (with $\mathcal{D} = \Sigma^l$, and $\tilde{K} = K_l$), Equation $(\star\star\star)$ and the fact that, by hypothesis, K_l is PSD. Consequently, K is PSD. □

4.6.2 AUC results for experiments on MHC-II

Calculating the area under the ROC curve (AUC)[Swets, 1988] for regression problems requires transforming the initial problem into a classification problem [Nielsen et al., 2008]. In the case of MHC-peptide binding affinities, the classification problem aims to distinguish between binders and non-binders. To achieve this, a threshold value allowing to distinguish binders from non-binders is required.

Single-target experiment

For this experiment, the predicted values were binding energies in kcal/mol. As proposed in Nielsen et al. [2008], we set a binding affinity threshold of 500nM. Therefore, the threshold

MHC β chain	AUC KRR+GS	AUC RTA	# of examples
DRB1*0101	**0.838**	0.749	5648
DRB1*0301	**0.781**	0.762	837
DRB1*0401	**0.753**	0.715	1014
DRB1*0404	0.786	**0.792**	617
DRB1*0405	**0.782**	0.757	642
DRB1*0701	**0.845**	0.790	833
DRB1*0802	0.724	**0.747**	557
DRB1*0901	0.665	**0.711**	551
DRB1*1101	**0.830**	0.753	812
DRB1*1302	0.708	**0.765**	636
DRB1*1501	**0.740**	0.736	879
DRB3*0101	0.716	**0.825**	483
DRB4*0101	**0.831**	0.799	664
DRB5*0101	**0.826**	0.732	835
H2*IA$_b$	**0.860**	0.828	526
H2*IA$_d$	0.772	**0.814**	306
Average :	**0.779**	0.767	

TABLE 4.6 – Results of the comparison between AUC values for binding energy predictions obtained from Kernel Ridge Regression and the GS Kernel versus the RTA [Bordner and Mittelmann, 2010b] method on the dataset proposed by the authors of this method.

value (t) in nanomolar was converted to kcal/mol using the technique proposed in Bordner and Mittelmann [2010b] :

$$t = -0.586 \times \log(500 \times 10^{-9}) = 8.50207 kcal/mol\,.$$

To calculate the AUC, we converted all the binding energies in the dataset to binary classes based on this binding threshold. Then, for all examples, we predicted the binding energy value (e) and generated a confidence value that the given example was a binder. This confidence value is given by : $c = e - t$ and then normalized using all other confidence values to be in the range $[0, 1]$. The latter were used to calculate the AUC for the experiment.

Pan-specific experiment

The AUC for this experiment was calculated using confidence values as explained above. A threshold of 500 nM was used to discriminate binders from non-binders [Nielsen et al., 2008].

MHC β chain	AUC KRR+GS	AUC MultiRTA	AUC NetMHCIIpan-2.0	# of examples
DRB1*0101	**0.807**	0.801	0.794	5166
DRB1*0301	0.775	0.751	**0.792**	1020
DRB1*0401	**0.802**	0.763	**0.802**	1024
DRB1*0404	0.862	0.835	**0.869**	663
DRB1*0405	**0.827**	0.808	0.823	630
DRB1*0701	**0.891**	0.817	0.886	853
DRB1*0802	0.840	0.786	**0.869**	420
DRB1*0901	**0.685**	0.674	0.684	530
DRB1*1101	**0.900**	0.819	0.875	950
DRB1*1302	0.648	**0.698**	0.648	498
DRB1*1501	**0.773**	0.729	0.769	934
DRB3*0101	0.690	**0.813**	0.733	549
DRB4*0101	0.759	0.746	**0.762**	446
DRB5*0101	**0.888**	0.788	0.879	924
Average :	0.796	0.773	**0.800**	

TABLE 4.7 – Results of the comparison between AUC values for binding affinity predictions obtained from Kernel Ridge Regression and the GS Kernel versus the MultiRTA [Bordner and Mittelmann, 2010a] and the NetMHCIIpan-2.0 [Nielsen et al., 2010] methods on the dataset proposed by the authors of NetMHCIIpan Nielsen et al. [2008].

Chapitre 5

Présentation du deuxième article

5.1 Détails de l'article

Titre :	MHC-NP : Predicting Peptides Naturally Processed by the MHC
Auteurs :	Sébastien Giguère, Alexandre Drouin, Alexandre Lacoste, Mario Marchand, Jacques Corbeil, François Laviolette
Lieu de publication :	Journal of Immunological Methods
Type de publication :	Article journal
Année :	2013

5.2 Contexte

Cet article fait suite à l'édition 2012 de la *Machine Learning Competition in Immunology* organisée par le *Dana-Farber Cancer Institute*. L'objectif de cette compétition était d'évaluer la capacité des approches par apprentissage automatique à prédire les antigènes naturellement présentés par le complexe majeur d'histocompatibilité (MHC) en surface des cellules. Les méthodes des participants ont été évaluées à l'aide de données non-publiées. Notre méthode a été jugée la plus précise. Suite à ces résultats, l'organisateur de la compétition, Dr. Vladimir Brusic, nous a recommandé de soumettre les détails de notre méthode dans le *Journal of Immunological Methods*. C'est donc cet article que nous présentons ici.

5.3 Contributions et discussion

Nous proposons un outil, MHC-NP, capable de prédire les antigènes présentés en surface des cellules par les complexes majeurs d'histocompatibilité. Le prédicteur est entrainé à l'aide de données fournies par le *Dana-Farber Cancer Institute*. Le problème de classification est en réalité un problème à trois classes contenant une hiérarchie. Il faut d'abord séparer les antigènes se liant au MHC de ceux ne se liant pas. Ces derniers ne pouvant être présentés en surface

puisque la molécule du MHC est centrale au mécanisme de présentation. Ensuite, parmi les antigènes se liant au MHC, certains possèdent des caractéristiques qui les rendent compatibles avec le mécanisme de présentation. Ce sont ces caractéristiques que nous apprenons par des techniques d'apprentissage automatique.

Nous présentons deux approches. La première fait abstraction de la hiérarchie inhérente entre les peptides qui lient le MHC et ceux présentés en surface de la cellule, alors que la seconde approche l'utilise. Nous démontrons qu'en exploitant cette hiérarchie, nous arrivons à améliorer la précision du prédicteur résultant. Nous comparons cette nouvelle approche à des approches de liaison peptide-MHC comme celle du chapitre précédent. Celles-ci simplifient le processus de présentation antigénique en un problème de liaison peptide-protéine. Cette comparaison nous permet de constater que l'approche que nous présentons surpasse les méthodes actuelles.

Notons qu'au Chapitre 8, nous présentons de nouveaux algorithmes d'apprentissage pour structures capables entre autres, de prédire l'appartenance d'un élément à une hiérarchie en exploitant les caractéristiques de celle-ci. Malgré que nous ne nous comparons pas à ces algorithmes ici, ceux-ci pourraient avoir d'importantes applications dans la prédiction des mécanismes biologiques comme celui étudié ici.

L'outil que nous avons développé est gratuit et est disponible sur le site web de l'*Immune Epitope Database*, la plus grosse base de données en immunologie, à l'adresse suivante : `http://tools.immuneepitope.org/mhcnp/`.

Chapitre 6

MHC-NP : Predicting peptides naturally processed by the MHC

We present MHC-NP, a tool for predicting peptides naturally processed by the MHC pathway. The method was part of the 2nd Machine Learning Competition in Immunology and yielded state-of-the-art accuracy for the prediction of peptides eluted from human HLA-A*02 :01, HLA-B*07 :02, HLA-B*35 :01, HLA-B*44 :03, HLA-B*53 :01, HLA-B*57 :01 and mouse H2-D^b and H2-K^b MHC molecules. We briefly explain the theory and motivations that have led to developing this tool. General applicability in the field of immunology and specifically epitope-based vaccine are expected. Our tool is freely available online and hosted by the Immune Epitope Database at `http://tools.immuneepitope.org/mhcnp/`.

6.1 Introduction

Epitope-based vaccines show great promise for diseases for which current approaches, such as pathogen attenuation, are not easily feasible or efficacious. Such vaccines have the advantage of being simpler to produce and induce a very specific immune response by targeting the immunogenic region of an antigen and specific MHC alleles of the host [Toussaint and Kohlbacher, 2009]. Computer assisted methods for the identification of immunogenic epitopes represent an important step in facilitating the creation of these next generation vaccines. Ideally, such vaccines could be adapted to target specific portions of the population, such as pregnant women and/or immunocompromised individuals, for which attenuated vaccines may present greater risks.

The major histocompatibility complex (MHC) is responsible for specific antigen recognition and inducing an appropriate cellular response. The MHC molecules are part of a complex pathway responsible for presenting antigens on the surface of antigen-presenting cells (APCs). Such antigens are presented under the form of MHC–peptide complexes, where the peptide

is a fragment of the antigen protein's sequence. Once MHC–peptide complexes are presented on an APC's surface, T cells recognise specific complexes and trigger an appropriate immune response. MHC molecules are categorised in two classes, MHC-class I, present antigens for CD8 T-cell driven responses and MHC-II, in contrast, present antigens for CD4 T-cell responses. Both pathways are complex and the binding of a peptide to a MHC molecule can be verified by in vitro or in silico methods.

Numerous approaches have been proposed for identifying MHC–peptide complexes [Zhang et al., 2012, Lundegaard et al., 2008, Giguère et al., 2013]. Most of these methods have focused on predicting the binding affinity of a given peptide and a MHC molecule. Unfortunately, the binding of a peptide and a MHC molecule is insufficient to ensure that the peptide will be processed to the surface of the cell. Indeed, only a subset of the peptides that bind to an MHC molecule can be naturally processed. Predicting such peptides is a difficult task.

In an effort to refine epitopic peptide identification methods, we propose a method to predict if a peptide is naturally processed by the MHC pathway. Our method has been trained on *in-vivo* and *in-vitro* data provided by D.K. Crockett and V. Brusic. The proposed method is based on statistical learning. Therefore, given that training data is available, it can be used for both MHC-I and MHC-II pathways. In addition, if used in conjunction with any binding affinity prediction tool, our method can validate if peptides can be processed by the MHC pathway. In the context of the 2012 Machine Learning Competition in Immunology (MLI), we have obtained empirical results which demonstrated that our method will have promising utility in immunology, vaccinology, and transplant rejection.

6.2 Material and methods

6.2.1 Data

The datasets used to train our method were those provided to the participants of the 2012 Machine Learning Competition in Immunology. These datasets comprised eight MHC-I molecules, composed of six human molecules (HLA-A*02 :01, HLA-B*07 :02, HLA-B*35 :01, HLA-B*44 :03, HLA-B*53 :01, and HLA-B*57 :01) and two mouse molecules (H2-D^b and H2-K^b). For each target molecule, three sets of peptides were provided : binding peptides, non-binding peptides and peptides obtained by elution. The peptides obtained by elution are naturally processed by the MHC-I pathway. Our method was tested using a set of data composed of binding, non-binding and naturally processed peptides. This testing data were provided by the organisers of the MLI competition and made publicly available `http://bio.dfci.harvard.edu/DFRMLI/HTML/natural.php`.

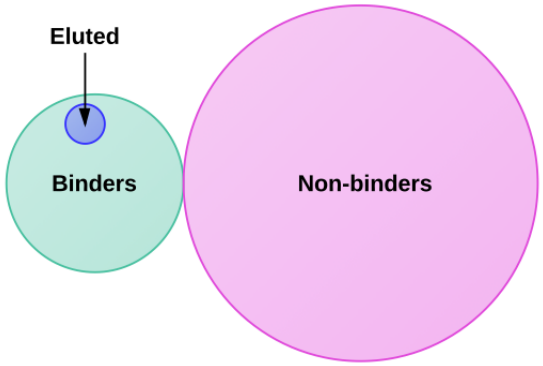


FIGURE 6.1 – An illustration of the different peptide classes contained in the datasets.

Discussion

From a machine learning point of view (see Hastie et al. [2005] for an introductory book), the training data were challenging in many ways. First, most learning algorithms receive data under the form of real valued feature vectors, although the training examples consisted of sequences of amino acids. Second, the length of training peptides varied from 8 to 11 amino acids. Most learning algorithms need to compute vector operations which are only defined for vectors of the same length. Therefore, any real valued vector representation of a peptide that depends on its length would ultimately fail. When facing such a problem, some authors have preferred to group peptides by length and to define independent learning problems for each length. This solution inevitably leads to inferior accuracy, since this severely reduces the number of training examples for each problem. The next section will introduce the concept of string kernels, which brings a solution to these two problems.

The particular case of identifying naturally processed peptides cannot be described as a typical binary or multiclass classification problem. Indeed, in this classification scheme, each example may only belong to one of the pre-defined classes. Yet, in our context, naturally processed peptides are all known to bind the MHC, whereas only $5-15\%$ of the binding peptides are naturally processed. Moreover, the problem has inherent noise in the data, due to the fact that a peptide can be a known binder, but not yet identified by elution. This implies that the eluted peptide class is a subset of the binding peptide class as illustrated in Figure 6.1. We therefore need to use a learning algorithm that is robust to noise and that accounts for the relationship between these classes.

Preprocessing the data

The provided data were obtained through experimental processes and, therefore, required a small amount of preprocessing to ensure that it contained no inconsistencies. First, for each MHC molecule, peptides that were simultaneously in the non-binding and binding sets

were discarded. Second, peptides that were simultaneously in the eluted and binding sets were removed from the binding set. Note that eluted peptides also bind to MHC molecules, although we preferred having the eluted peptides in a distinct set. These two steps ensured that the three sets of peptides were disjoint and therefore that the peptides comprised in each set shared one common characteristic (eluted-binder, non-eluted-binder, non-binder). We will respectively refer to these three sets of peptides as "E", "B" and "N".

6.2.2 Learning approach

Machine learning is a method of choice for building predictive software when facing complex phenomena. Classification is the task of assigning one of possibly many pre-defined classes to each example produced by a phenomena of interest. In this paper, we will restrict ourselves to the case of binary classification, where each example can only belong to two classes.

A learning algorithm should minimize the task loss. Here, the task is to distinguish naturally processed peptides from all other peptides (non-eluted-binders and non-binders). Therefore, while developing our method, we have focussed on learning a predictor for peptides that are naturally processed by the MHC pathway. Consequently, we have only considered the following two classes : peptides that are naturally processed (E) and peptides that are not naturally processed (B and N). This predictor was presented to the MLI 2012 competition, ultimately yielding excellent prediction accuracy. Throughout this paper, we will refer to this model as (E vs BN).

In an attempt to further improve this method, we have, since the MLI competition, developed an alternate model. This model consists in learning two predictors : a predictor to distinguish binding peptides (E and B) from non-binding peptides (N) and a predictor to distinguish eluted peptides (E) from binding peptides (B). The prediction is done in a decision tree fashion : the (EB vs N) predictor is first applied, then, only if the peptide is predicted to be a binder, the (E vs B) predictor is applied to determine if the peptide is naturally processed. Throughout this paper, we will refer to this model as (EB vs N + E vs B).

For comparison purposes, a binding/non-binding predictor was learned to distinguish binding peptides (E + B) from non-binding peptides (N). This predictor purposely does not distinguish naturally processed peptides from binding peptides. Such a predictor allows to establish a direct comparison to current tools, which rely on MHC-peptide binding affinity. This predictor also allows to measure the capability of our method to discriminate naturally processed peptides from binding peptides.

Learning algorithm

To learn the predictors described in the previous section, we have used the soft-margin support vector machine (SVM) algorithm originally proposed by Cortes and Vapnik [1995]. This

supervised learning algorithm is considered to be the state of the art for classification problems.

The input of the SVM algorithm consists in a set of pairs. For each of these pairs, the first element is an example and the second element is a class that has been attributed to the example by an expert. First, each learning example is mapped to a possibly high dimensional vector space called the feature space. Then, the SVM finds a separating hyperplane that splits the feature space in two halves, such that examples from the same class lie on the same side. Also, the hyperplane is such that all learning examples have the greatest Euclidian distance (also known as the geometric margin) from it. The soft margin SVM provides a parameter allowing a trade-off between the number of classification errors and the size of the geometric margin on the correctly classified examples. This allows SVM to tolerate that certain examples of a given class be on the wrong side of the hyperplane and thus prevent the hyperplane from being affected by noise in the data. This parameter is generally tuned by cross-validation techniques that are described in Section 6.3.2.

To perform predictions, new testing examples are first mapped to the same space as the training examples. Then, their classes are assigned depending on their location with respect to the hyperplane. The greater the geometric margin for an example, the more confident the predictor is about its predicted class. This notion of margin can be further transformed into a probability estimate using techniques such as the Platt scaling [Platt, 1999]. Our proposed tool : MHC-NP, outputs, for each new peptide, a probability estimate of the peptide being naturally processed by the specified MHC pathway.

From sequences to feature vectors

The kernel trick [Shawe-Taylor and Cristianini, 2004] is a method to map complex structures, such as strings, to a high dimensional vector space in which the dot product is defined. A kernel function is a function that implicitly computes the dot product in this high dimensional space, without explicitly mapping the structures to this space. Such functions therefore avoid the prohibitive computational cost of computing dot products in high dimensional vector spaces. Recall that every algorithm that constructs a hyperplane heavily depends on dot product operations. In addition, as for dot products, kernels can be interpreted as similarity functions. The more similar two examples are, the greater the kernel function value. Kernel functions have been proposed for a variety of biological structures including protein primary structure [Saigo et al., 2004] and protein tertiary structure [Qiu et al., 2007].

Many machine learning algorithms like the SVM have been kernelized by substituting the dot product operation of their objective and prediction functions by kernel function evaluations. Such kernelized algorithms can work with complex data, such as strings, given that there exists a kernel defined for this data type. Kernels that compute the similarity between two strings are called string kernels. There is no doubt that string kernels have been partly responsible

for the increased use of machine learning algorithms in biology.

The generic string kernel

As highlighted in the previous section, string kernels are similarity functions that can be used with kernelized learning algorithms. A common approach is to compare strings by their common k-mers for a chosen value of k. For example,"LFQLITA" and "LFQRPPLI" can be split into their 2-mers, which are respectively ("LF", "FQ", "QL", "LI", "TA") and ("LF", "FQ", "QR", "RP", "PP", "PL", "LI"). Then, the similarity value between these two sequences is given by the number of common 2-mers. Here, the similarity is 3, since the sequences have "LF", "FQ", and "LI" as common 2-mers. This similarity function corresponds to the Spectrum kernel [Leslie et al., 2002]. Note that this kernel allows the comparison of strings of different lengths and corresponds to a dot product in a high dimensional vector space. Therefore, such kernels alleviate the need for considering peptides of different lengths separately and allow better usage of the data.

Comparing peptides using k-mer is a simple approach that gives a broad estimate of their similarity. However, for most proteins, notably for MHC, peptide–protein interactions occur at a very specific location known as the binding site. The relative position of residues in this binding site and their physicochemical properties are key aspects that drive the interaction. Nevertheless, the Spectrum kernel does not account for these two important biological features.

In an effort to consider the relative position of residues in similarity function, Meinicke et al. [2004] proposed the Oligo kernel, which was the first string kernel to account for the relative position of k-mers. Instead of counting the number of common k-mer, the Oligo kernel assigns a weight to each common k-mer depending on their relative positions in the two peptides. For example, in the peptides "LFQLITA" and "LFQRPPLI", the 2-mers "LF" and "FQ" share the same position. In contrast, the 2-mer "LI" is respectively at position 4 and 7. For this reason, the contribution of "LF" and "FQ" to the similarity should be greater than the contribution of "LI". In this sense, the author of the Oligo kernel proposed to weight the contribution of common k-mers by using a function inversely proportional to their distance in the peptides.

Moreover, in an attempt to account for the physicochemical properties of the residues in the binding site, Toussaint et al. [2010] proposed to weight the contribution of k-mers as function of their physicochemical properties. This is based on the fact that an amino acid in a peptide can sometimes be substituted by another amino acid with similar properties, such as hydrophobicity, charge or molecular weight, without affecting the peptide's binding affinity. Toussaint and collaborators proposed to incorporate such knowledge in the kernel function by using the physicochemical properties of their amino acids to compare k-mers.

Inspired by the ideas of Meinicke et al. [2004] and Toussaint et al. [2010], Giguère et al. [2013] proposed the generic string (GS) kernel. This kernel accounts for the physicochemical

properties and the relative position of amino acids in the comparison of k-mers. The GS kernel was shown to outperform state-of-the-art prediction methods on single-target and pan-specific peptide-MHC-II binding affinity prediction benchmark datasets and three Quantitative Structure Affinity Model benchmark datasets. Giguère et al. [2013] have also proposed a dynamic programming algorithm for the fast computation of their kernel and have shown that the GS kernel induces a dot product in a high dimensional vector space. The GS kernel has four parameters, namely, two for controlling the importance of comparing the physico-chemical properties of amino acids, one for setting the maximum length of k-mers and one controlling the penalty enquired due to the relative distance of k-mers. As described in the next section, these parameters can be tuned by cross-validation.

Implementation details

In order to ensure reproducibility, all experimentations were conducted using the SVM implementation of the Scikit-Learn library [Pedregosa et al., 2011] and the GS kernel [Giguère et al., 2013], available at `http://graal.ift.ulaval.ca/gs-kernel/`, both free and open source softwares.

6.3 Theory/calculation

6.3.1 Assessing model performance

To assess to prediction accuracy of the different approaches, we used the area under the ROC curve (AUC) and the F1 score [Bradley, 1997]. In addition, we have used the sensitivity and the specificity to analyse the performance of our method.

The Receiver Operating Characteristic (ROC) curve provides a graphical illustration of a predictor's recall and specificity by plotting the recall and $(1 - specificity)$ as a function of the threshold. This type of curve can be used to select a threshold with respect to some trade-off between recall and specificity. The AUC is a threshold independent metric obtained by computing the area under the ROC curve. This metric is closely related to the one used to assess the performance of the methods submitted to the 2012 MLI competition, which is given by

$$\sqrt{Sensitivity} + \sqrt{Specificity} + Sensitivity \cdot 10^{-5}\,. \tag{6.1}$$

Indeed, the organizer of the competition ranked the methods by selecting the threshold maximizing Equation 6.1, thus making their metric threshold independent.

In the field of machine learning, the accuracy is a threshold dependent metric that is used to evaluate prediction methods. Unfortunately, the datasets used to train our method are

unbalanced, which means that the number of eluted peptide is much smaller than the number of non-eluted peptides. Thus, using the accuracy to evaluate the performance of our method could be misleading. For example, if only 5% of the peptides contained in a dataset were eluted, a predictor could abstain from predicting any peptide as being eluted and would nevertheless achieve an accuracy of 95%. For this reason, we propose to use the F1 score to evaluate our method's discriminative power. The F1 score is given by Equation 6.2 and its value is comprised between 0 and 1. This metric has the advantage of being unaffected by class imbalance.

$$F1\ \ score = 2 \cdot \frac{precision \cdot sensitivity}{precision + sensitivity} \tag{6.2}$$

It is important to mention that small changes in the decision threshold can lead to important differences in threshold dependent metrics such as the F1 score. However, it is crucial to compare the ability of a method to estimate a good threshold using the training data. Without a threshold, a method can only provide a confidence score to indicate how confident it is about an example belonging to a target class. Although, for most real world applications, a binary decision must be made. Thus, a threshold is required to distinguish between a positive and a negative decision. Moreover, recent learning paradigms, such as transductive learning [Joachims, 1999], domain adaptation [Jiang, 2008] and positive and unlabelled learning [Elkan and Noto, 2008] aim at learning models specifically designed for a target task by using abundant unlabelled data. These new approaches have been shown to outperform supervised learning when few labelled training examples are available. The additional discrimination power of such methods could be undetected by using only threshold independent metrics. To ensure that improvements made by future methods are discernable, we recommend using threshold dependent metrics such as the F1 score.

6.3.2 Algorithm parameter selection

For all models, 10-fold cross-validation (see Hastie et al. [2005]) on the training set was used for selecting the SVM and the GS kernel parameters. All the metric values reported in the Results and discussion section were computed on the (independent) testing set provided by the organisers of the 2012 MLI competition.

6.3.3 Advanced parameter tuning

In the cross-validation method, a single parameter is chosen to produce a good predictor with a limited amount of data. This parameter is estimated based on the ability of the algorithm to yield an accurate predictor. However, with a limited amount of data, the uncertainties in the estimations tell us that predictors obtained by using other parameters values should also be considered. In this case, Bayesian theory suggests to use a (probabilistic) combination of many predictors obtained with different parameter values [Lacoste et al., 2014]. Such a combination

MHC allele	AUC E vs BN	AUC (EB vs N+E vs B)	F1 score E vs BN	F1 score (EB vs N+E vs B)
HLA-A*02 :01	0.8573	**0.8806**	0.4114	**0.4795**
HLA-B*07 :02⋆	0.9157	**0.9236**	0.5644	**0.5992**
HLA-B*35 :01	0.9187	**0.9367**	0.6720	**0.7059**
HLA-B*44 :03	**0.8178**	0.7947	**0.5833**	0.5443
HLA-B*53 :01⋆	0.8401	**0.8515**	**0.6368**	0.5758
HLA-B*57 :01	0.8017	**0.8258**	**0.6623**	0.6447
H2-D^b⋆	**0.8614**	0.8437	0.3125	**0.3724**
H2-K^b⋆	0.8185	**0.8251**	0.3212	**0.3421**
Average	0.8539	**0.8602**	0.5205	**0.5330**

TABLE 6.1 – Comparison of the two eluted peptide prediction models. For each allele, the best results are shown in bold. Alleles followed by a star symbol are those for which our method performed the best in the MLI competition.

allows to take into account the uncertainty on determining which predictor is truly the most accurate. Elaborated repeated resampling techniques allow to estimate the probability of each predictor being the most accurate. These probabilities are then used to weight the contribution of each predictor in the final decision.

In the context of MLI, a preliminary version of this method was used, yielding outstanding results for the HLA-B*07:02, H2-D^b and H2-K^b alleles. However, additional experiments showed that this approach does not significantly improve the AUC or F1 scores.

6.4 Results and discussion

6.4.1 Selecting a model for eluted peptide prediction

In this paper, we have presented two eluted peptide prediction models : the one predictor approach (E vs BN) and the two predictor approach (EB vs N + E vs B). In order to determine which model is the most accurate, we computed their AUC and F1 score on the testing set of each allele. The results on all eight alleles are shown in Table 6.1. For the AUC and F1 score results, we observe that the (EB vs N + E vs B) approach outperforms the (E vs BN) one on six out of eight alleles. Unfortunately, the number of alleles was insufficient to compute p-values, therefore no statistical analysis of the results was made. Nevertheless, the (EB vs N + E vs B) method clearly promises better prediction accuracy than the (E vs BN) method.

This result is interesting from a machine learning point of view, since it indicates that it is better to handle the binding affinity prediction task and the eluted peptide prediction task separately.

Both methods have parameters that are tuned by cross-validation to fit closely to the prediction task. Among the most important parameters to tune, are those of the kernel function which effectively define a specialized similarity function. Generally, theses parameters vary greatly depending on the learning task.

The two predictor approach (EB vs N + E vs B) was tuned twice, once to distinguish binding from non-binding peptides, and a second time, to isolate eluted peptides. The optimal parameters found for each task were considerably different. This result was expected, since both tasks are significantly different and, thus, require different similarity functions. We believe that this is one of the main reasons why the two predictor approach outperforms the single predictor approach. In addition, note that the former exploits the structure of the problem by taking into account that eluted peptides are also binders.

Due to its superior accuracy, we have chosen to use the (EB vs N + E vs B) method to elaborate our eluted peptide prediction tool, MHC-NP. This tool is hosted by the Immune Epitope Database [Peters et al., 2005] and publicly available at `http://tools.immuneepitope.org/mhcnp/`.

6.4.2 Comparison to binding affinity prediction methods

The purpose of the 2012 Machine Learning Competition in Immunology was to assess the ability of computational methods for predicting peptides naturally processed by the MHC-I pathway. It is known that there exists a correlation between the binding of peptides and their immunogenicity [Toussaint and Kohlbacher, 2009]. Therefore, strong binders are more likely to be naturally processed by the MHC pathway. For this reason, we have chosen to compare our most accurate eluted peptide predictor, the (EB vs N + E vs B) approach, to two state-of-the-art MHC-I binding affinity prediction methods. The first method, called (EB vs N), is inspired by the peptide–protein binding affinity work of [Giguère et al., 2013]. Note that this method is equivalent to performing the first prediction task of the (EB vs N + E vs B) approach. This comparison allows to estimate the additional discriminative power that follows from using a (E vs B) predictor in combination with a binding affinity predictor. For the sake of completeness, we also compare our approach to the popular NetMHC-3.2 [Lundegaard et al., 2008], which was used as a benchmark method in the 2012 MLI competition.

Table 6.2 shows AUC results on the testing sets for the three methods. Our approach outperforms both the (EB vs N) approach and NetMHC-3.2 on five out of eight alleles. Also, MHC-NP achieves an average AUC of 0.8602, which is greater than both binding affinity prediction methods who respectively obtained 0.8454 and 0.8265.

As seen in Table 6.3, our method outperforms the EB vs N approach and NetMHC-3.2 on five out of eight alleles. Our approach also achieved the best average F1 score : 0.5330, in comparison to 0.4983 and 0.4324 respectively obtained by the EB vs N and the NetMHC-3.2

MHC Allele	MHC-NP (EB vs N + E vs B)	EB vs N	NetMHC-3.2
HLA-A*02 :01	0.8806	0.9078	**0.9310**
HLA-B*07 :02⋆	**0.9236**	0.9075	0.9042
HLA-B*35 :01	**0.9367**	0.9307	0.9090
HLA-B*44 :03	0.7947	0.7778	**0.8104**
HLA-B*53 :01⋆	**0.8515**	0.7817	0.6651
HLA-B*57 :01	0.8258	**0.8438**	0.8181
H2-D^b⋆	**0.8437**	0.8031	0.7641
H2-K^b⋆	**0.8251**	0.8106	0.8098
Average	**0.8602**	0.8454	0.8265

TABLE 6.2 – Comparison of the MHC-NP (EB vs N + E vs B) eluted peptide prediction method and two binding affinity prediction methods using the area under the ROC curve. For each allele, the best result is shown in bold. Alleles followed by a star symbol are those for which our method performed the best in the MLI competition.

MHC Allele	MHC-NP (EB vs N + E vs B)	EB vs N	NetMHC-3.2
HLA-A*02 :01	0.4795	0.4452	**0.5833**
HLA-B*07 :02⋆	**0.5992**	0.4791	0.5239
HLA-B*35 :01	0.7059	**0.7432**	0.7417
HLA-B*44 :03	**0.5443**	0.4655	0.4697
HLA-B*53 :01⋆	**0.5758**	0.5030	0.4568
HLA-B*57 :01	0.6447	**0.7085**	0.3006
H2-D^b⋆	**0.3724**	0.3008	0.0833
H2-K^b⋆	**0.3421**	0.3408	0.3000
Average	**0.5330**	0.4983	0.4324

TABLE 6.3 – Comparison of the MHC-NP (EB vs N + E vs B) eluted peptide prediction method and two binding affinity prediction methods using the F1 score. For each allele, the best result is shown in bold. Alleles followed by a star symbol are those for which our method performed the best in the MLI competition.

method.

Considering our results, it is unclear why, for some alleles, the simpler approach of predicting MHC–peptide binding affinity outperforms the MHC-NP method at predicting eluted peptides. Part of the 2012 MLI competition was to assess if the latter task was learnable. Our results show that, for most alleles, this task can be learned with good accuracy. The performance of the NetMHC-3.2 tool for the HLA-A*02 :01 allele and the (EB vs N) approach for the HLA-B*35 :01 and HLA-B*57 :01 alleles could be attributed to noise in the naturally

MHC Allele	Sensitivity	Specificity
HLA-A*02 :01	0.5000	0.9589
HLA-B*07 :02	0.5259	0.9805
HLA-B*35 :01	0.5926	0.9795
HLA-B*44 :03	0.4725	0.9618
HLA-B*53 :01	0.5229	0.9149
HLA-B*57 :01	0.6504	0.8091
H2-D^b	0.2784	0.9868
H2-K^b	0.2301	0.9925
Average	0.4716	0.9480

TABLE 6.4 – Sensitivity and specificity of the MHC-NP (EB vs N + E vs B) method on all alleles. Alleles followed by a star symbol are those for which our method performed the best in the MLI competition.

processed peptides data.

Finally, Table 6.4 reports the sensitivity and specificity of MHC-NP on all alleles. The proposed approach achieves an average sensitivity of 0.4716 and an impressive average specificity of 0.948. Recall that sensitivity and specificity depend on a decision threshold. Considering the high specificity and low sensitivity of MHC-NP, it is reasonable to think that the decision threshold of MHC-NP could be selected to allow a better specificity/sensitivity trade-off. Given the AUC results reported in Table 6.2, such a threshold clearly exists.

6.5 Conclusion

We proposed a new method, MHC-NP, for the prediction of peptides naturally processed by the MHC pathway. We showed that MHC-NP outperforms state-of-the-art approaches based on MHC binding affinity. Moreover, the results support the hypothesis that peptides that strongly bind the MHC molecule have greater propensity of being naturally processed to the cell surface. In absence of eluted peptide data, the prediction of MHC binding affinity remains a reasonable approach to identify naturally processed peptides. Furthermore, the proposed approach is amenable to be used in conjunction with state-of-the-art pan-specific MHC binding tools to improve its prediction accuracy. Finally, the approach could be further improved by using additional data on molecules that contribute to the MHC pathway.

6.6 Acknowledgements

We thank G.L. Zhang, V. Brusic and their collaborators for organising the MLI 2012 competition and providing the data. We thank the Immune Epitope Database for hosting MHC-NP.

Computations were performed on the GPC supercomputer at the SciNet HPC Consortium. SciNet is funded by : the Canada Foundation for Innovation under the auspices of Compute Canada ; the Government of Ontario ; Ontario Research Fund - Research Excellence ; and the University of Toronto. We also thank Calcul Quebec and Laval University for their support. JC acknowledges the support of the Canada Research Chair in Medical Genomics. This work was supported in part by the Fonds de recherche du Québec - Nature et technologies (FL, MM & JC : 2013-PR-166708) and the NSERC Discovery Grants (FL : 262067, MM : 122405).

Chapitre 7

Présentation du troisième article

7.1 Détails de l'article

Titre :	Risk Bounds and Learning Algorithms for the Regression Approach to Structured Output Prediction
Auteurs :	Sébastien Giguère, François Laviolette, Mario Marchand, Khadidja Sylla
Lieu de publication :	International Conference on Machine Learning (ICML)
Type de publication :	Article de conférence
Année :	2013

Titre :	PAC-Bayesian Risk Bounds and Learning Algorithms for the Regression Approach to Structured Output Prediction
Auteurs :	Sébastien Giguère, François Laviolette, Mario Marchand, Amélie Rolland
Lieu de publication :	Advanced Structured Prediction, MIT Press
Type de publication :	Chapitre d'un livre
Année :	2014

7.2 Contexte

Cette contribution a originalement été publiée lors de l'*International Conference on Machine Learning* (ICML). Suite à cette visibilité, les éditeurs du livre *Advanced Structured Prediction* nous ont approchés pour participer à l'écriture de celui-ci. Nous avons donc bonifié et amélioré l'article de la conférence pour en faire un chapitre dans ce livre. Pour ces raisons, la version présentée dans cette thèse est celle du livre *Advanced Structured Prediction*.

7.3 Contributions et discussion

De nombreux algorithmes de prédiction de structures, comme les réseaux de Markov à marge maximale et les machines à vecteurs de supports structurés, doivent résoudre le problème, souvent NP-difficile, de pré-image lors de l'apprentissage et de l'inférence. L'approche de régression à valeurs vectorielles a l'avantage d'éviter le problème de pré-image, au moins au cours de l'apprentissage. Dans ce chapitre, nous fournissons des garanties rigoureuses pour cette approche. Plus spécifiquement, nous montrons que la perte quadratique est un substitut à la perte de prédiction lorsque le noyau de sortie satisfait certaines conditions par rapport à la perte de prédiction. De plus, nous montrons empiriquement, que la perte quadratique peut partiellement remplacer la pré-image lors de la sélection des hyper-paramètres. Nous présentons deux bornes supérieures du risque de prédiction qui dépendent du risque quadratique empirique du prédicteur. Le minimiseur de la première borne est le prédicteur proposé par Cortes et al. [2007] alors que le minimiseur de la seconde est nouveau [Giguère et al., 2013b]. Les deux prédicteurs sont comparés sur deux tâches pratiques : la prédiction de mots manuscrits et la classification hiérarchique des propriétés enzymatiques de protéines. Les deux prédicteurs donnent des précisions comparables, mais similaires à l'état de l'art.

En somme, ce chapitre apporte une meilleure compréhension des algorithmes de prédiction de structures. Bien que ceux-ci soient encore peu utilisés, nous sommes convaincus qu'ils auront plusieurs applications en biologie computationnelle. Entre autre, un des problèmes utilisé pour évaluer ces algorithmes consiste à classifier l'activité de protéines dans un arbre de propriétés enzymatiques. Déjà, l'outil du Chapitre 6 pour la prédiction des antigènes présentés en surface des cellules par les MHCs pourrait bénéficier de ces mêmes avancements.

Comme les algorithmes que nous présentons sont à base de noyaux, ils ont l'avantage d'être indépendants de la structure des exemples d'apprentissage. Par exemple, l'approche utilisée pour la prédiction des mots manuscrits pourrait être utilisée pour prédire d'autres types de séquences biologiques comme la structure secondaire de protéines.

Bien que l'approche par régression permette d'éviter le problème de pré-image durant l'apprentissage et la sélection des hyper-paramètres, ce problème complexe doit toujours être résolu à l'étape de prédiction. Actuellement, peu de noyaux sont connus pour avoir une pré-image facile à calculer. Nous croyons que la recherche de nouveaux noyaux spécialement conçus pour avoir une pré-image facilement calculable et le développement d'algorithmes d'optimisation combinatoire pour résoudre la pré-image des noyaux existants, permettra d'étendre significativement les champs d'applications des algorithmes de prédiction de structures.

Au Chapitre 10, nous présentons un algorithme de faible complexité qui permet de maximiser la fonction d'inférence de la régression de ridge lorsque le noyau est le *Generic String*. Bien qu'il ne s'agit pas strictement du problème de pré-image, celui que nous explorons dans ce chapitre en est très proche. Déjà, nous avons obtenu des résultats préliminaires démontrant

l'applicabilité de l'algorithme du Chapitre 10 pour résoudre la pré-image du *Generic String*.

Finalement, notons que la première borne que nous présentons s'applique également à la régression de ridge à noyau, l'algorithme de régression que nous utilisons au Chapitre 4 et 10. En effet, celle-ci borne le risque quadratique lorsque la sortie est vectorielle. La régression de ridge étant simplement un cas particulier où la dimensionnalité du vecteur de sortie est unitaire.

Chapitre 8

PAC-Bayesian Risk Bounds and Learning Algorithms for the Regression Approach to Structured Output Prediction

Many structured output prediction algorithms, such as max-margin Markov networks and the structural SVM, need to solve the (often NP-hard) pre-image problem during training and inference. The vector-valued regression approach has the computational advantage of avoiding this problem, at least, during training. In this chapter, we provide rigorous guarantees for this approach. More specifically, we show that the quadratic regression loss is a convex surrogate of the prediction loss when the output kernel satisfies some condition with respect to the prediction loss. We then present two upper bounds of the prediction risk that depend on the empirical quadratic risk of the predictor. The minimizer of the first bound is the predictor proposed by Cortes et al. [2007]. The minimizer of the second bound was recently proposed by Giguère et al. [2013b]. Both predictors are compared on practical tasks, yielding similar, but state of the art accuracies.

8.1 Introduction

Structured output prediction is a supervised learning problem where the goal of the learner is to predict the correct output y associated to some given input x. Here, the output y can be a complex object such as a sequence of symbols, a parse tree, or a graph. The predictor generally consists of a vector w of real-valued weights and each input-output example (x, y) is mapped to a high-dimensional feature vector $\boldsymbol{\phi}(x, y)$. The output predicted by w on input x is then the output y that maximizes the inner product $\langle w, \boldsymbol{\phi}(x, y)\rangle$. However, as emphasized

by Gärtner and Vembu [2009], this pre-image problem is often NP-hard. Consequently, any learning algorithm that needs to solve this pre-image problem, for several weight vectors and every training example, will often take a prohibitive running time. This is probably the most important problem facing several state-of-the-art structured output learning algorithms such as max-margin Markov networks [Taskar et al., 2004] and the structural SVM [Tsochantaridis et al., 2005].

One of the first attempts to design a learning algorithm that avoids the pre-image problem is due to Cortes et al. [2007]. They have proposed to find the predictor that minimizes an ℓ_2-regularized vector-valued regression objective (which does not depend on the predicted output for a given input). Also, they have obtained empirical results that compare favorably to those of structural SVM and max-margin Markov networks on the word-recognition data set used by Taskar et al. [2004]. In this chapter, we present guarantees for such a regression approach by first showing that the quadratic loss function used by Cortes et al. [2007] provides a convex upper bound on the original prediction loss (that depends on the predicted output) provided that the output kernel satisfies some condition with respect to the prediction loss. We also present two PAC-Bayes upper bounds [1] for the prediction risk that depend on the quadratic empirical loss used by Cortes et al. [2007]. These two bounds have been first proposed by Giguère et al. [2013b], the paper on which this chapter is largely based. The minimizer of the first bound turns out to be the same as the one proposed by Cortes et al. [2007] while the minimizer of the second bound, valid for arbitrary reproducing kernel Hilbert spaces (RKHS), has been proposed by Giguère et al. [2013b]. Both predictors are compared on practical tasks.

PAC-Bayes theory has also been applied recently [McAllester, 2007] to structured output prediction for a stochastic predictor that aims at minimizing the expected prediction risk. The resulting learning algorithms need to solve the pre-image problem on each example of the training set for each update of the predictor. In contrast, we present here risk bounds for learning algorithms that avoid solving the pre-image problem and that produce a deterministic predictor instead of a stochastic one.

8.2 From Structured Output Prediction to Vector-Valued Regression

In the supervised learning setting, the learner has access to a set $S \stackrel{\text{def}}{=} \{(x_1, y_1), \ldots, (x_m, y_m)\}$ of m training examples where each example consists of an input-output pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$. The input space $\mathcal{X}$ and the output space $\mathcal{Y}$ are both arbitrary but we assume the existence of both an input feature map $\phi_{\mathcal{X}} : \mathcal{X} \to \mathcal{H}_{\mathcal{X}}$ and an output feature map $\phi_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{H}_{\mathcal{Y}}$, where both $\mathcal{H}_{\mathcal{X}}$ and $\mathcal{H}_{\mathcal{Y}}$ are high-dimensional vector spaces and, more generally, reproducing kernel

1. See [McAllester, 2003, Langford, 2005] or [Germain et al., 2009] for an introductory text about PAC-Bayesian theory

Hilbert spaces (RKHS). In $\mathcal{H}_\mathcal{Y}$, we use $\langle \boldsymbol{\phi}_\mathcal{Y}(y), \boldsymbol{\phi}_\mathcal{Y}(y') \rangle$ to denote the inner product and use $\|\boldsymbol{\phi}_\mathcal{Y}(y)\|^2 \stackrel{\text{def}}{=} \langle \boldsymbol{\phi}_\mathcal{Y}(y), \boldsymbol{\phi}_\mathcal{Y}(y) \rangle$ for the squared norm. The analogue notation is used in $\mathcal{H}_\mathcal{X}$.

Given access to a training set S, the task of the learner is to construct a structured predictor which is represented by a linear operator $\mathbf{W}$ that transforms vectors of $\mathcal{H}_\mathcal{X}$ into vectors of $\mathcal{H}_\mathcal{Y}$. For any $x \in \mathcal{X}$ and any $\mathbf{W}$, the output $y_\mathbf{W}(x)$ predicted by $\mathbf{W}$ is given by

$$y_\mathbf{W}(x) \stackrel{\text{def}}{=} \operatorname*{argmin}_{y \in \mathcal{Y}} \left\| \boldsymbol{\phi}_\mathcal{Y}(y) - \mathbf{W}\boldsymbol{\phi}_\mathcal{X}(x) \right\| . \tag{8.1}$$

Note that $y_\mathbf{W}(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \langle \boldsymbol{\phi}_\mathcal{Y}(y), \mathbf{W}\boldsymbol{\phi}_\mathcal{X}(x) \rangle$ whenever $\|\boldsymbol{\phi}_\mathcal{Y}(y)\|$ is the same for all $y \in \mathcal{Y}$. In this case, we recover the usual structured output prediction method when the joint feature vectors $\boldsymbol{\phi}(x, y)$ are tensor products $\boldsymbol{\phi}_\mathcal{X}(x) \otimes \boldsymbol{\phi}_\mathcal{Y}(y)$. Since finding $y_\mathbf{W}(x)$ given x and $\mathbf{W}$ is generally NP-hard [Gärtner and Vembu, 2009], we want to avoid solving this pre-image problem.

We consider feature maps that are defined by kernels such that $K_\mathcal{Y}(y, y') = \langle \boldsymbol{\phi}_\mathcal{Y}(y), \boldsymbol{\phi}_\mathcal{Y}(y') \rangle$ for all $(y, y') \in \mathcal{Y}^2$ and $K_\mathcal{X}(x, x') = \langle \boldsymbol{\phi}_\mathcal{X}(x), \boldsymbol{\phi}_\mathcal{X}(x') \rangle$ for all $(x, x') \in \mathcal{X}^2$. We will see that the proposed solutions for $\mathbf{W}$ will have the property that $\mathbf{W}\boldsymbol{\phi}_\mathcal{X}(x) = \sum_{i=1}^m \sum_{j=1}^m \boldsymbol{\phi}_\mathcal{Y}(y_i) A_{i,j} K_\mathcal{X}(x_j, x)$ for some $m \times m$ matrix $\mathbf{A}$. Consequently, the predicted output $y_\mathbf{W}(x)$ only requires the use of the kernels $K_\mathcal{X}$ and $K_\mathcal{Y}$ (instead of the feature maps $\phi_\mathcal{X}$ and $\phi_\mathcal{Y}$).

Note that this class of predictors $\mathbf{W} : \mathcal{H}_\mathcal{X} \to \mathcal{H}_\mathcal{Y}$ is strictly included in the class of predictors studied by Brouard et al. [2011] and Kadri et al. [2013], where the input kernel has values $K_\mathcal{X}(x, x')$ that are operators : $\mathcal{H}_\mathcal{Y} \to \mathcal{H}_\mathcal{Y}$. It is still unknown how the risk bounds developed in this chapter can be extended to this more general case.

We assume that each example (x, y) is generated independently according to some unknown distribution D. Given a function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that quantifies the loss incurred on (x, y) when the predicted output is $y_\mathbf{W}(x)$, the task of the learner is to find the predictor that minimizes the expected loss (or risk) $\mathbf{E}_{(x,y)\sim D} L(y_\mathbf{W}(x), y)$. We refer to L as the *prediction loss*.

Note that the output kernel $K_\mathcal{Y}$, being a similarity measure on $\mathcal{Y}^2$, induces a loss function $L_{K_\mathcal{Y}}$ defined as

$$\begin{aligned} L_{K_\mathcal{Y}}(y_\mathbf{W}(x), y) &\stackrel{\text{def}}{=} \frac{1}{2}\|\boldsymbol{\phi}_\mathcal{Y}(y) - \boldsymbol{\phi}_\mathcal{Y}(y_\mathbf{W}(x))\|^2 \\ &= \frac{K_\mathcal{Y}(y, y) + K_\mathcal{Y}(y_\mathbf{W}(x), y_\mathbf{W}(x))}{2} - K_\mathcal{Y}(y, y_\mathbf{W}(x)) . \end{aligned} \tag{8.2}$$

We refer to $L_{K_\mathcal{Y}}$ as the *output kernel loss*.

Both the prediction loss and the output kernel loss on (x, y) depend on the predicted output $y_\mathbf{W}(x)$. This is in sharp contrast with the *quadratic loss* $\left\| \boldsymbol{\phi}_\mathcal{Y}(y) - \mathbf{W}\boldsymbol{\phi}_\mathcal{X}(x) \right\|^2$ which does not depend on $y_\mathbf{W}(x)$. However we can show that the quadratic loss provides an upper bound to the output kernel loss.

Lemma 3. *For any structured predictor* $\mathbf{W}$ *giving predictions as defined by Equation* (8.1)*, for any* $(x, y) \in \mathcal{X} \times \mathcal{Y}$*, we have*

$$L_{K_\mathcal{Y}}(y_\mathbf{W}(x), y) \;\leq\; 2 \left\| \boldsymbol{\phi}_\mathcal{Y}(y) - \mathbf{W}\boldsymbol{\phi}_\mathcal{X}(x) \right\|^2 .$$

Proof. From the triangle inequality, we have, for all $\mathbf{W}$ and for all (x, y),

$$\|\boldsymbol{\phi}_\mathcal{Y}(y) - \boldsymbol{\phi}_\mathcal{Y}(y_\mathbf{W}(x))\| \leq \|\boldsymbol{\phi}_\mathcal{Y}(y) - \mathbf{W}\boldsymbol{\phi}_\mathcal{X}(x)\| + \|\boldsymbol{\phi}_\mathcal{Y}(y_\mathbf{W}(x)) - \mathbf{W}\boldsymbol{\phi}_\mathcal{X}(x)\| .$$

From Equation (8.1), we have $\|\boldsymbol{\phi}_\mathcal{Y}(y_\mathbf{W}(x)) - \mathbf{W}\boldsymbol{\phi}_\mathcal{X}(x)\| \;\leq\; \|\boldsymbol{\phi}_\mathcal{Y}(y) - \mathbf{W}\boldsymbol{\phi}_\mathcal{X}(x)\|$ for all $\mathbf{W}$ and for all (x, y). Hence, from these two inequalities, we have $\|\boldsymbol{\phi}_\mathcal{Y}(y) - \boldsymbol{\phi}_\mathcal{Y}(y_\mathbf{W}(x))\| \leq\; 2\|\boldsymbol{\phi}_\mathcal{Y}(y) - \mathbf{W}\boldsymbol{\phi}_\mathcal{X}(x)\|$, which gives the lemma. □

Lemma 3 has far-reaching consequences whenever we use an output kernel $K_\mathcal{Y}$ such that $L(y, y') \leq L_{K_\mathcal{Y}}(y, y')$ for all $(y, y') \in \mathcal{Y}^2$ because, in that case, we have

$$L(y_\mathbf{W}(x), y) \;\leq\; 2 \left\| \boldsymbol{\phi}_\mathcal{Y}(y) - \mathbf{W}\boldsymbol{\phi}_\mathcal{X}(x) \right\|^2 ,$$

for all predictors $\mathbf{W}$ and all $(x, y) \in \mathcal{X} \times \mathcal{Y}$.

Under these circumstances, a predictor $\mathbf{W}$ having a small quadratic risk $\mathbf{E}_{(x,y)\sim D} \left\| \boldsymbol{\phi}_\mathcal{Y}(y) - \mathbf{W}\boldsymbol{\phi}_\mathcal{X}(x) \right\|^2$ has also a small prediction risk

$$\mathbf{E}_{(x,y)\sim D} L(y_\mathbf{W}(x), y) .$$

To minimize the prediction risk, we need to solve the (usually hard) pre-image problem of finding the predicted output $y_\mathbf{W}(x)$ for every example in the training set and for all predictors $\mathbf{W}$ tried by the learning algorithm. Thanks to Lemma 3, we can avoid this computational burden by performing the simpler regression task of minimizing the quadratic risk whenever we use an output kernel $K_\mathcal{Y}$ for which the output kernel loss $L_{K_\mathcal{Y}}$ upper bounds the prediction loss L.

Consider the case when the prediction loss L is the zero-one loss. In that case any output kernel $K_\mathcal{Y}$ for which there exists two different outputs y and y' having $K_\mathcal{Y}(y, y) = K_\mathcal{Y}(y', y') = K_\mathcal{Y}(y, y')$ will not give an output kernel loss $L_{K_\mathcal{Y}}$ which upper bounds L. But the Dirac kernel for which $K_\mathcal{Y}(y, y') = 1$ if $y = y'$ and 0 otherwise gives an $L_{K_\mathcal{Y}}$ which is identical to L.

In the case where the prediction loss L is the Hamming distance, the Hamming kernel (given by the length of the largest string minus the Hamming distance between the two strings) provides an output structured loss $L_{K_\mathcal{Y}}$ identical to L. One could also use any output kernel giving an $L_{K_\mathcal{Y}}$ which upper bounds the Hamming distance at the expense of introducing an additional slackness between the quadratic risk and the prediction risk.

A predictor achieving a small quadratic risk also achieves a small prediction risk when the output kernel $K_\mathcal{Y}$ gives an $L_{K_\mathcal{Y}}$ which upper bounds L. However, there exist data-generating

distributions where the predictor achieving the smallest possible quadratic risk has a substantially larger prediction risk than the predictor achieving the smallest possible prediction risk. In other words, there is no consistency guarantee for the regression approach to structured output prediction because no such guarantee exists for the particular case of binary classification [2]. However, the regression approach avoids the computational burden of dealing with the pre-image problem and, under some distributions, there might be some kernels for which there exists predictors achieving a small quadratic risk.

Thanks to Lemma 3, any upper bound on the quadratic risk also provides a bound on the prediction risk (provided that there exists an output kernel loss that upper bounds the prediction loss). Consequently, the upper bounds proposed by Caponnetto and De Vito [2007] and Baldassarre et al. [2012] also provide bounds on the prediction risk for predictors minimizing the ℓ_2-regularized least-squares. However, instead of focussing explicitly on such predictors, we provide bounds that hold simultaneously for any predictor $\mathbf{W}$ and that depend on the empirical quadratic risk achieved by $\mathbf{W}$ on the training data.

8.3 A PAC-Bayesian Bound with Isotropic Gaussians

Values of the prediction loss $L(y_{\mathbf{w}}(x), y)$ are always between zero and one. However, this is clearly not the case for the quadratic loss $\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2$. Theoretically attainable very large loss values are well known to give very loose concentration inequalities and, unavoidably, very large risk bounds. Therefore, to obtain a tighter risk bound, we use the following lemma which upper bounds the prediction loss in terms of a bounded function of the quadratic loss.

Lemma 4. *For any prediction loss L upper-bounded by the output kernel loss $L_{K_{\mathcal{Y}}}$, for any (x, y), any $\mathbf{W}$, and any $a \geq 1$, we have*

$$L(y_{\mathbf{w}}(x), y) \;\leq\; \frac{ae}{e-1}\left(1 - e^{-\frac{2}{a}\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}\right) .$$

Proof. For any $0 \leq x \leq 1$, we have $x \leq \frac{e}{e-1}\left(1 - e^{-x}\right)$. Therefore

$$\begin{aligned}\frac{1}{a}L(y_{\mathbf{w}}(x), y) &\leq \frac{e}{e-1}\left(1 - e^{-\frac{1}{a}L(y_{\mathbf{w}}(x), y)}\right) \\ &\leq \frac{e}{e-1}\left(1 - e^{-\frac{2}{a}\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}\right) ,\end{aligned}$$

where the last equality follows from Lemma 3 and the fact that L is upper-bounded by $L_{K_{\mathcal{Y}}}$. □

2. Indeed, it is easy to find distributions for which the minimizer of the quadratic risk gives a classifier which achieves a much larger 0-1 risk than the optimal classifier. See the Appendix for a simple example.

8.3.1 The Risk Bound

We propose here an upper bound on the prediction risk that uses PAC-Bayes theory to upper bound $\mathbf{E}_{(x,y)\sim D}\left(1 - e^{-\frac{2}{a}\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}\right)$ for some $a \geq 0$. However, PAC-Bayes theory does not directly provide bounds on deterministic predictors such as $\mathbf{W}$. Instead, it provides guarantees for stochastic Gibbs predictors that are described in terms of a posterior distribution Q over deterministic predictors. More precisely, PAC-Bayes theory provides bounds for the Gibbs risk defined as the Q-average of the risk of deterministic predictors. The following theorem, due to Zhang [2006], provides an example of such a bound [3].

Theorem 5. *(from Zhang [2006]) Let ζ be any loss function, and let P be any prior distribution on $\mathcal{W}$, the set of all the predictors $\mathbf{W} : \mathcal{H}_{\mathcal{X}} \to \mathcal{H}_{\mathcal{Y}}$. Then, for any D on $\mathcal{X} \times \mathcal{Y}$, with probability at least $1 - \delta$ over all training sets S sampled according to D^m, we have, simultaneously for all distributions Q on $\mathcal{W}$,*

$$-\mathop{\mathbf{E}}_{\mathbf{v}\sim Q} \ln \mathop{\mathbf{E}}_{(x,y)\sim D} e^{-\zeta(\mathbf{v},x,y)} \leq$$
$$\frac{1}{m}\left(\mathop{\mathbf{E}}_{\mathbf{v}\sim Q} \sum_{i=1}^{m} \zeta(\mathbf{v}, x_i, y_i) + \mathrm{KL}(Q, P) + \ln\frac{1}{\delta}\right),$$

where $\mathrm{KL}(Q, P)$ denotes the Kullback-Leibler divergence between distributions Q and P.

To use Theorem 5, we restrict ourselves (in this section) to the case where both feature spaces $\mathcal{H}_{\mathcal{X}}$ and $\mathcal{H}_{\mathcal{Y}}$ are finite-dimensional vector spaces of dimensions $N_{\mathcal{X}}$ and $N_{\mathcal{Y}}$ respectively. The set of predictors thus coincides with the set of $N_{\mathcal{Y}} \times N_{\mathcal{X}}$ matrices. Each posterior is chosen to be an isotropic Gaussian of variance σ^2 and expectation $\mathbf{W}$. If $Q_{\mathbf{W},\sigma}(\mathbf{V})$ denotes the density at a matrix $\mathbf{V}$ of this posterior, we have

$$Q_{\mathbf{W},\sigma}(\mathbf{V}) \;=\; \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^{N_{\mathcal{X}}N_{\mathcal{Y}}} e^{-\frac{1}{2\sigma^2}\|\mathbf{V}-\mathbf{W}\|^2}, \tag{8.3}$$

where, for any matrix $\mathbf{V}$, $\|\mathbf{V}\|^2 \stackrel{\text{def}}{=} \sum_{i=1}^{N_{\mathcal{Y}}}\sum_{j=1}^{N_{\mathcal{X}}} V_{i,j}^2$ (also called the Frobenius norm of $\mathbf{V}$).

For the prior P, we chose the (non-informative) isotropic Gaussian centered at the origin, *i.e.*, $P = Q_{\mathbf{0},\sigma}$. In that case, we have

$$\mathrm{KL}(Q_{\mathbf{W},\sigma}, P) \;=\; \frac{1}{2}\frac{\|\mathbf{W}\|^2}{\sigma^2}. \tag{8.4}$$

The next theorem provides an upper bound on the risk of the (deterministic) predictor $\mathbf{W}$ which depends on its empirical quadratic risk—not on the empirical risk of a stochastic (Gibbs) predictor. This new result was made possible by performing Gaussian integrals over functions

3. Unfortunately, there is no dependence on the sample-size m in the theorem stated by Zhang [2006] because the one-example formulation was used. We obtain Theorem 5 if we use m examples instead of one.

of the quadratic loss and by observing that we can choose a value for σ such that the noise of the empirical quadratic risk is cancelled by the noise of the true quadratic risk whenever $K_{\mathcal{X}}(x,x)$ is the same for all $x \in \mathcal{X}$.

Theorem 6. *Consider any input kernel $K_{\mathcal{X}}$ and any output kernel $K_{\mathcal{Y}}$ inducing finite-dimensional feature spaces. Suppose that $K_{\mathcal{X}}(x,x) = 1$ for all $x \in \mathcal{X}$. Let D be any distribution on $\mathcal{X} \times \mathcal{Y}$. Then, for any prediction loss L upper-bounded by the output kernel loss $L_{K_{\mathcal{Y}}}$, with probability at least $1-\delta$ over all training sets S sampled according to D^m, we have, simultaneously for all predictors $\mathbf{W}$,*

$$\mathop{\mathbf{E}}_{(x,y)\sim D} L(y_{\mathbf{W}}(x), y) \leq \frac{5e}{e-1}\left[1 - e^{-\frac{1}{m}\left(2\sum_{i=1}^{m}\|\boldsymbol{\phi}_{\mathcal{Y}}(y_i) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x_i)\|^2 + \frac{9}{8}\|\mathbf{W}\|^2 + \ln\frac{1}{\delta}\right)}\right].$$

Proof. If we use Theorem 5 in the case of the quadratic loss with the proposed posterior $Q_{\mathbf{W},\sigma}$ and prior P, and if we use Equations (8.4) and (8.5) and exploit the convexity of $-\ln x$, we then have that, with probability at least $1-\delta$,

$$-\ln\left(\mathop{\mathbf{E}}_{(x,y)\sim D}\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} e^{-2\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}\right) \leq \frac{1}{m}\left(\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} 2\sum_{i=1}^{m}\|\boldsymbol{\phi}_{\mathcal{Y}}(y_i) - \mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x_i)\|^2 + \frac{\|\mathbf{W}\|^2}{2\sigma^2} + \ln\frac{1}{\delta}\right).$$

In the Appendix we provide proofs of the following Gaussian integrals :

$$\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 = \|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 + \sigma^2 N_{\mathcal{Y}}\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2. \tag{8.5}$$

$$\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} e^{-2\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2} = \left[\frac{\sigma^{N_{\mathcal{X}}}}{\sqrt{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}}\right]^{N_{\mathcal{Y}}} e^{-\frac{2\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}}. \tag{8.6}$$

Since, by hypothesis, $\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|$ is a constant independent of x, with probability at least $1-\delta$,

$$\mathop{\mathbf{E}}_{(x,y)\sim D} 1 - e^{-2\frac{\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2\sigma^2}} \leq 1 - e^{-\xi - \frac{1}{m}\left(2\sum_{i=1}^{m}\|\boldsymbol{\phi}_{\mathcal{Y}}(y_i) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x_i)\|^2 + \frac{\|\mathbf{W}\|^2}{2\sigma^2} + \ln\frac{1}{\delta}\right)}, \tag{8.7}$$

where

$$\xi \stackrel{\text{def}}{=} N_{\mathcal{Y}}\left[2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2\sigma^2 + \ln\left(\frac{\sigma^{N_{\mathcal{X}}}}{\sqrt{1+4\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2\sigma^2}}\right)\right].$$

For $\|\boldsymbol{\phi}_{\mathcal{X}}(x)\| = 1$, the value of σ^2 satisfying $\xi = 0$ is monotonously increasing with $N_{\mathcal{X}}$; going from $\sigma^2 = 0,6752...$ for $N_{\mathcal{X}} = 1$ to $\sigma^2 = 1$ when $N_{\mathcal{X}} \to \infty$. Consider Inequality (8.7) when $\xi = 0$. In that case $2/3 < \sigma^2 \leq 1$. Then its right-hand side can be upper-bounded by the same quantity but with σ^2 replaced by $2/3$, and its left-hand side can be lower-bounded by the same quantity but with σ^2 replaced by 1. The theorem then follows by applying Lemma 4 for $a = 5$. □

8.3.2 The Risk Bound Minimizer

The predictor $\mathbf{W}$ that minimizes the risk bound of Theorem 6 is the one that minimizes the multiple-output ridge regression objective F_{rr}, where

$$F_{rr}(\mathbf{W}) \stackrel{\text{def}}{=} C \sum_{i=1}^{m} \|\boldsymbol{\phi}_{\mathcal{Y}}(y_i) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x_i)\|^2 + \|\mathbf{W}\|^2 ,$$

for some value of $C > 0$. Note that F_{rr} is exactly the objective to minimize that was proposed by Cortes et al. [2007]. At optimality, the gradient of F_{rr} must vanish. As shown by Cortes et al. [2007], the solution $\mathbf{W}^*$ is unique for finite C and is given by

$$\mathbf{W}^* = \sum_{i=1}^{m}\sum_{j=1}^{m} \boldsymbol{\phi}_{\mathcal{Y}}(y_i)(\mathbf{K}_{\mathcal{X}} + \frac{1}{C}\mathbf{I})^{-1}_{i,j}\boldsymbol{\phi}_{\mathcal{X}}^{\top}(x_j) , \tag{8.8}$$

where $\boldsymbol{\phi}_{\mathcal{X}}^{\top}(x)$ denotes the transpose of vector $\boldsymbol{\phi}_{\mathcal{X}}(x)$, $\mathbf{K}_{\mathcal{X}}$ denotes the input kernel matrix, and $\mathbf{I}$ denotes the $m \times m$ identity matrix.

Since $\mathbf{W}^*$ is the minimizer of the ℓ_2-regularized least squares F_{rr}, the convergence rates established by Caponnetto and De Vito [2007] also apply to $\mathbf{W}^*$.

8.4 A Sample Compressed PAC-Bayesian Bound

Note that the predictor minimizing the ridge regression objective is a linear combination of simple predictors $\boldsymbol{\phi}_{\mathcal{Y}}(y_i)\boldsymbol{\phi}_{\mathcal{X}}^{\top}(x_j)$ that are identified by two training examples. Inspired by some recent work on PAC-Bayes sample-compression [Laviolette and Marchand, 2007, Germain et al., 2011], we want to establish a guarantee on the true risk for arbitrary linear combinations of these simple structured output predictors. In contrast with Theorem 6, the obtained risk bound will be valid for feature spaces $\mathcal{H}_{\mathcal{X}}$ and $\mathcal{H}_{\mathcal{Y}}$ that are arbitrary RKHS (of possibly infinite dimensionality). For that purpose, let $\boldsymbol{\phi}_{\mathcal{X}}^{\dagger}(x)$ denote the dual of vector $\boldsymbol{\phi}_{\mathcal{X}}(x)$. The dual $\boldsymbol{\phi}_{\mathcal{X}}^{\dagger}(x)$ is a map from $\mathcal{H}_{\mathcal{X}}$ to $\mathbb{R}$ such that for all $(x, x') \in \mathcal{X}^2$ we have $\boldsymbol{\phi}_{\mathcal{X}}^{\dagger}(x)\boldsymbol{\phi}_{\mathcal{X}}(x') = \langle \boldsymbol{\phi}_{\mathcal{X}}(x), \boldsymbol{\phi}_{\mathcal{X}}(x')\rangle = K_{\mathcal{X}}(x, x')$. Thus, given a training set S of m examples, we consider predictors that can be written as

$$\mathbf{W} = \sum_{i=1}^{m}\sum_{j=1}^{m} \boldsymbol{\phi}_{\mathcal{Y}}(y_i) A_{i,j} \boldsymbol{\phi}_{\mathcal{X}}^{\dagger}(x_j) , \tag{8.9}$$

where $A_{i,j} \in \mathbb{R}$ for all $(i, j) \in \{1, \ldots, m\}^2$. In this case, the quadratic loss $\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2$ is now given by

$$\left\| \boldsymbol{\phi}_{\mathcal{Y}}(y) - \sum_{i=1}^{m}\sum_{j=1}^{m} A_{i,j} K_{\mathcal{X}}(x_j, x) \boldsymbol{\phi}_{\mathcal{Y}}(y_i) \right\|^2 \stackrel{\text{def}}{=} R(\mathbf{A}, x, y) . \tag{8.10}$$

To connect with PAC-Bayes sample-compression, let us write $\mathbf{A}$ in terms of a distribution $\mathbf{q}$ over $2m^2$ predictors. For this purpose, let $q^+_{i,j} \geq 0$ be the weight on predictor $\boldsymbol{\phi}_{\mathcal{Y}}(y_i)\boldsymbol{\phi}_{\mathcal{X}}^{\dagger}(x_j)$

and let $q^-_{i,j} \geq 0$ be the weight on the opposite predictor $-\boldsymbol{\phi}_{\mathcal{Y}}(y_i)\boldsymbol{\phi}^{\dagger}_{\mathcal{X}}(x_j)$ such that

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{s\in\{-1,+1\}} q^s_{i,j} = 1\,.$$

Now, without loss of generality, for all (i,j), let $A_{i,j} = \kappa\cdot(q^+_{i,j} - q^-_{i,j})$ for some $\kappa > 0$. For notational brevity, let $R(\mathbf{q},x,y)$ be the quadratic loss obtained from $R(\mathbf{A},x,y)$ when each $A_{i,j}$ is replaced by $\kappa\cdot(q^+_{i,j} - q^-_{i,j})$. In addition, let $\mathcal{I} \stackrel{\text{def}}{=} \{1,\ldots,m\}^2$ denote the set of all pairs of indices and let $\mathcal{B} \stackrel{\text{def}}{=} \{-1,+1\}$. We then have

$$R(\mathbf{q},x,y) \;=\; \sum_{\mathbf{i}\in\mathcal{I}}\sum_{\mathbf{j}\in\mathcal{I}}\sum_{s\in\mathcal{B}}\sum_{t\in\mathcal{B}} q^s_{\mathbf{i}}\,q^t_{\mathbf{j}}\,\ell^{s,t}_{\mathbf{i},\mathbf{j}}(x,y)\,,$$

where, for $\mathbf{i} = (i,i')$ and $\mathbf{j} = (j,j')$, we have

$$\ell^{s,t}_{\mathbf{i},\mathbf{j}}(x,y) \stackrel{\text{def}}{=} \langle \boldsymbol{\phi}_{\mathcal{Y}}(y) - \kappa s\boldsymbol{\phi}_{\mathcal{Y}}(y_i)K_{\mathcal{X}}(x_{i'},x), \boldsymbol{\phi}_{\mathcal{Y}}(y) - \kappa t\boldsymbol{\phi}_{\mathcal{Y}}(y_j)K_{\mathcal{X}}(x_{j'},x)\rangle\,. \tag{8.11}$$

An upper bound on $R(\mathbf{q}) \stackrel{\text{def}}{=} \mathbf{E}_{(x,y)\sim D}R(\mathbf{q},x,y)$ also provides an upper bound on the prediction risk $\mathbf{E}_{(x,y)\sim D}L(y_{\mathbf{w}}(x),y)$ since, by Lemma 3, we have $L(y_{\mathbf{w}}(x),y) \leq 2R(\mathbf{q},x,y)$ whenever $A_{i,j}$ is replaced by $\kappa\cdot(q^+_{i,j} - q^-_{i,j})$ in Equation (8.9) and whenever the L is upper-bounded by $L_{K_{\mathcal{Y}}}$. Our goal is thus to find a tight upper bound on $R(\mathbf{q})$ and then design an algorithm that finds $\mathbf{q}$ (hence, the predictor $\mathbf{W}$) that minimizes this upper bound.

8.4.1 The Risk Bound

The proposed risk bound follows from PAC-Bayes theory and depends on how far is the posterior distribution $\mathbf{q}$ from a prior $\mathbf{p}$. For $\mathbf{p}$, we choose the uniform distribution over $\mathcal{I} \stackrel{\text{def}}{=} \{1,\ldots,2m\}^2$ so that $p^s_{\mathbf{i}} = 1/(2m^2)$ for all $(\mathbf{i},s) \in \mathcal{I}\times\mathcal{B}$, where $\mathcal{B} \stackrel{\text{def}}{=} \{-1,+1\}$. The posterior $\mathbf{q}$ is chosen to be *quasi-uniform*. By this we mean that for all $\mathbf{i}\in\mathcal{I}$ we have $q^+_{\mathbf{i}} + q^-_{\mathbf{i}} = 1/m^2$. In that case, each $q^s_{\mathbf{i}} \in [0, 1/m^2]$ and, consequently, the KL-divergence $\mathrm{KL}(\mathbf{q},\mathbf{p})$ is always at most $\ln 2$. Such a small upper bound on $\mathrm{KL}(\mathbf{q},\mathbf{p})$ contributes significantly at reducing the risk bound closer to the empirical risk $R(\mathbf{q},S) \stackrel{\text{def}}{=} (1/m)\sum_{i=1}^m R(\mathbf{q},x_i,y_i)$. Moreover, restricting $\mathbf{q}$ to quasi-uniform distributions does not restrict the class of predictors considered by the learner. Indeed, for any predictor $\mathbf{W}$ described by some matrix $\mathbf{A}$ in Equation (8.9), there exists $\kappa > 0$ and a quasi-uniform $\mathbf{q}$ such that $A_{i,j} = \kappa\cdot(q^+_{i,j} - q^-_{i,j})$.

Theorem 7. *Let $a \leq \ell^{s,t}_{\mathbf{i},\mathbf{j}}(x,y) \leq b$ for all $(x,y)\in\mathcal{X}\times\mathcal{Y}$, for all $(s,t)\in\mathcal{B}^2$, for all $(\mathbf{i},\mathbf{j})\in\mathcal{I}^2$, and for some interval $[a,b]$. Let D be any distribution on $\mathcal{X}\times\mathcal{Y}$. Let $m \geq 8$. Then, with probability at least $1-\delta$ over all training sets S sampled according to D^m, we have, simultaneously for all quasi-uniform distributions $\mathbf{q}$ on $\mathcal{I}$,*

$$R(\mathbf{q}) \;\leq R(\mathbf{q},S) + \sqrt{\frac{b-a}{2(m-4)}\left[20 + \ln\left(\frac{8\sqrt{m}}{\delta}\right)\right]}\,.$$

Proof. Given the uniform prior $\mathbf{p}$, consider the Laplace transform

$$\mathcal{L}_{\mathbf{p}} \stackrel{\text{def}}{=} \sum_{\mathbf{i}\in\mathcal{I}}\sum_{\mathbf{j}\in\mathcal{I}}\sum_{s\in\mathcal{B}}\sum_{t\in\mathcal{B}} p_{\mathbf{i}}^s p_{\mathbf{j}}^t \exp\left(2(m-4)\left(\frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}(S)-a}{b-a}-\frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}-a}{b-a}\right)^2\right),$$

where $\ell_{\mathbf{i},\mathbf{j}}^{s,t} \stackrel{\text{def}}{=} \mathbf{E}_{(x,y)\sim D}\ell_{\mathbf{i},\mathbf{j}}^{s,t}(x,y)$ and $\ell_{\mathbf{i},\mathbf{j}}^{s,t}(S) \stackrel{\text{def}}{=} (1/m)\sum_{i=1}^m \ell_{\mathbf{i},\mathbf{j}}^{s,t}(x_i,y_i)$. Note that $\ell_{\mathbf{i},\mathbf{j}}^{s,t}(S)$ is a biased estimate of $\ell_{\mathbf{i},\mathbf{j}}^{s,t}$ since it considers the loss on examples that are used for the predictors described by $(\mathbf{i},s)$ and $(\mathbf{j},t)$. To obtain an unbiased estimator, let $S_{\mathbf{i},\mathbf{j}} \stackrel{\text{def}}{=} \{(x_i,y_i)\in S : i\notin \mathbf{i}\cup\mathbf{j}\}$ and let $m_{\mathbf{i},\mathbf{j}} \stackrel{\text{def}}{=} |S_{\mathbf{i},\mathbf{j}}|$. Then, let[4] $\ell_{\mathbf{i},\mathbf{j}}^{s,t}(S_{\mathbf{i},\mathbf{j}}) \stackrel{\text{def}}{=} \frac{1}{m_{\mathbf{i},\mathbf{j}}}\sum_{k=1}^m I((x_k,y_k)\in S_{\mathbf{i},\mathbf{j}})\ell_{\mathbf{i},\mathbf{j}}^{s,t}(x_k,y_k)$ be our unbiased estimator of $\ell_{\mathbf{i},\mathbf{j}}^{s,t}$. It is then straightforward to show that $\ell_{\mathbf{i},\mathbf{j}}^{s,t}(S_{\mathbf{i},\mathbf{j}}) - 4(b-a)/m \le \ell_{\mathbf{i},\mathbf{j}}^{s,t}(S) \le \ell_{\mathbf{i},\mathbf{j}}^{s,t}(S_{\mathbf{i},\mathbf{j}}) + 4(b-a)/m$ and, consequently, for $m\ge 8$

$$\left(\frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}(S)-a}{b-a}-\frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}-a}{b-a}\right)^2 \le \left(\frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}(S_{\mathbf{i},\mathbf{j}})-a}{b-a}-\frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}-a}{b-a}\right)^2 + \frac{10}{m}. \tag{8.12}$$

Now, given $\mathrm{kl}(q,p) \stackrel{\text{def}}{=} q\ln(q/p) + (1-q)\ln[(1-q)/(1-p)]$, if we use $2(q-p)^2 \le \mathrm{kl}(q,p)$, we obtain

$$\begin{aligned}
\mathcal{L}_{\mathbf{p}} \le & \mathop{\mathbf{E}}_{S\sim D^m} \sum_{\mathbf{i}\in\mathcal{I}}\sum_{\mathbf{j}\in\mathcal{I}}\sum_{s\in\mathcal{B}}\sum_{t\in\mathcal{B}} \\
& p_{\mathbf{i}}^s p_{\mathbf{j}}^t \exp\left(\frac{20}{m}(m-4) + m_{\mathbf{i},\mathbf{j}}\mathrm{kl}\left(\frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}(S_{\mathbf{i},\mathbf{j}})-a}{b-a}, \frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}-a}{b-a}\right)\right) \\
= & \exp\left(\frac{20}{m}(m-4)\right)\sum_{\mathbf{i}\in\mathcal{I}}\sum_{\mathbf{j}\in\mathcal{I}}\sum_{s\in\mathcal{B}}\sum_{t\in\mathcal{B}} p_{\mathbf{i}}^s p_{\mathbf{j}}^t \left(\prod_{i\in\mathbf{i}\cup\mathbf{j}} \mathop{\mathbf{E}}_{(x_i,y_i)\sim D}\right) \\
& \mathop{\mathbf{E}}_{S_{\mathbf{i},\mathbf{j}}\sim D^{m_{\mathbf{i},\mathbf{j}}}} \exp\left(m_{\mathbf{i},\mathbf{j}}\mathrm{kl}\left(\frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}(S_{\mathbf{i},\mathbf{j}})-a}{b-a}, \frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}-a}{b-a}\right)\right).
\end{aligned}$$

Since $S_{\mathbf{i},\mathbf{j}}$ is the arithmetic mean of $m_{\mathbf{i},\mathbf{j}}$ i.i.d. random variables, the lemma of Maurer [2004] tells us that the last expectation (over $S_{\mathbf{i},\mathbf{j}}$) is at most $2\sqrt{m_{\mathbf{i},\mathbf{j}}}$ and, consequently, $\mathcal{L}_{\mathbf{p}} \le 2\sqrt{m}\exp(20(m-4)/m)$. Since $\mathcal{L}_{\mathbf{p}}$ is the expectation (over S) of a positive random variable, we can use Markov's inequality which states that, with probability of at least $1-\delta$ over the random draws of S, we have

$$\begin{aligned}
& \ln\left(\sum_{\mathbf{i}\in\mathcal{I}}\sum_{\mathbf{j}\in\mathcal{I}}\sum_{s\in\mathcal{B}}\sum_{t\in\mathcal{B}} p_{\mathbf{i}}^s p_{\mathbf{j}}^t \exp\left(2(m-4)\left(\frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}(S)-a}{b-a}-\frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}-a}{b-a}\right)^2\right)\right) \\
& \le \frac{20}{m}(m-4) + \ln\left(\frac{2\sqrt{m}}{\delta}\right).
\end{aligned}$$

By turning the expectation over $\mathbf{p}^2$ into an expectation over $\mathbf{q}^2$, and by using Jensen's inequality on the concavity of the logarithm, the last inequality implies that we have

$$\sum_{\mathbf{i}\in\mathcal{I}}\sum_{\mathbf{j}\in\mathcal{I}}\sum_{s\in\mathcal{B}}\sum_{t\in\mathcal{B}} q_{\mathbf{i}}^s q_{\mathbf{j}}^t \left(\frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}(S)-a}{b-a}-\frac{\ell_{\mathbf{i},\mathbf{j}}^{s,t}-a}{b-a}\right)^2 \le \frac{1}{2(m-4)}\left(\mathrm{KL}(\mathbf{q}^2,\mathbf{p}^2) + 20 + \ln\frac{2\sqrt{m}}{\delta}\right),$$

4. Here $I(a)=1$ if predicate a is true and $I(a)=0$ otherwise.

for all $\mathbf{q}$. The theorem then follows by using Jensen's inequality on the convexity of $(q-p)^2$ and by using $\mathrm{KL}(\mathbf{q}^2, \mathbf{p}^2) = 2\mathrm{KL}(\mathbf{q}, \mathbf{p}) \leq 2\ln 2$ for quasi-uniform posteriors. □

Hence, for quasi-uniform posteriors $\mathbf{q}$, the upper bound on $R(\mathbf{q})$ is very close to $R(\mathbf{q}, S)$ whenever $(b-a) \ll m$. From Equation (8.11), we can see that $(b-a)$ is at most $2B_{\mathcal{Y}}(1+\kappa B_{\mathcal{X}})^2$ when $|K_{\mathcal{X}}(x,x')| \leq B_{\mathcal{X}}$ for all $(x,x') \in \mathcal{X}^2$ and $|K_{\mathcal{Y}}(y,y')| \leq B_{\mathcal{Y}}$ for all $(y,y') \in \mathcal{Y}^2$.

8.4.2 The Risk Bound Minimizer

The posterior $\mathbf{q}$ that minimizes the upper bound of Theorem 7 is the posterior minimizing $R(\mathbf{q}, S)$ under the constraint that $\mathbf{q}$ is quasi-uniform. In that case, each $q^s_{i,j} \in [0, 1/m^2]$. Since $A_{i,j} = \kappa \cdot (q^+_{i,j} - q^-_{i,j})$, the quasi-uniform constraint on $\mathbf{q}$ implies that $|A_{i,j}| \leq C$ for all $(i,j) \in \mathcal{I}$ and for some $C > 0$. Instead of handling these m^2 constraints, it is computationally much cheaper to replace them by the single ℓ_2 constraint $\sum_{(i,j)\in\mathcal{I}} A^2_{i,j} \leq R^2$ for some $R > 0$. Note that, although this is an approximate solution, we have $|A_{i,j}| \leq R$ for all (i,j) whenever this ℓ_2 constraint is satisfied. Hence, given $R^2 \stackrel{\text{def}}{=} m^2\rho^2$ and $R(\mathbf{A}, S) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^m R(\mathbf{A}, x_i, y_i)$, let us solve

$$\begin{aligned} &\min_{\mathbf{A}} \; R(\mathbf{A}, S) \\ &\text{s.t.} \; \sum_{i=1}^m \sum_{j=1}^m A^2_{i,j} \leq R^2 \,. \end{aligned} \tag{8.13}$$

Theorem 8. *Let $\mathcal{A}^*$ denote the set of solutions of problem (8.13). Let $\mathbf{K}_{\mathcal{X}}$ and $\mathbf{K}_{\mathcal{Y}}$ denote, respectively, the input and output kernel matrices. Let $v_1, \ldots, v_m$ and $\lambda_1, \ldots, \lambda_m$ denote, respectively, the eigenvectors and eigenvalues of $\mathbf{K}_{\mathcal{X}}$. Let $u_1, \ldots, u_m$ and $\delta_1, \ldots, \delta_m$ denote, respectively, the eigenvectors and eigenvalues of $\mathbf{K}_{\mathcal{Y}}$. Let $\mathcal{J} \stackrel{def}{=} \{(i,j) \in \mathcal{I} : \delta_i\lambda_j > 0\}$. Then*

$$\sum_{i=1}^m \sum_{j=1}^m \gamma_{i,j} u_i v_j^\top \in \mathcal{A}^* \,,$$

where $\gamma_{i,j}$ is given by

$$\gamma_{i,j} = \begin{cases} 0 & \text{if } \delta_i\lambda_j = 0 \\ \frac{u_i^\top v_j}{\lambda_j} & \text{if } \delta_i\lambda_j > 0 \end{cases} \qquad \text{if } \sum_{(i,j)\in\mathcal{J}} \frac{(u_i^\top v_j)^2}{\lambda_j^2} \leq R^2 \,,$$

$$\gamma_{i,j} = \frac{\delta_i\lambda_j(u_i^\top v_j)}{\delta_i\lambda_j^2 + m\beta} \qquad \text{otherwise} \,,$$

where $\beta > 0$ is the solution of

$$\sum_{i=1}^m \sum_{j=1}^m \frac{\delta_i^2\lambda_j^2(u_i^\top v_j)^2}{(\delta_i\lambda_j^2 + m\beta)^2} \;=\; R^2 \,.$$

Proof. Let $L(\mathbf{A}, \beta) \stackrel{\text{def}}{=} R(\mathbf{A}, S) + \beta \left(\|\mathbf{A}\|^2 - R^2 \right)$. Convex optimisation theory tells us that if there exists $\beta \geq 0$ and $\mathbf{A}$ with $\|\mathbf{A}\|^2 \leq R^2$, that satisfies $\partial L / \partial \mathbf{A} = 0$ and $\beta \cdot \left(\|\mathbf{A}\|^2 - R^2 \right) = 0$, then $\mathbf{A} \in \mathcal{A}^*$. Here we have

$$\frac{\partial L}{\partial \mathbf{A}} \;=\; \frac{2}{m} \mathbf{K}_{\mathcal{Y}} (\mathbf{A}\mathbf{K}_{\mathcal{X}} - \mathbf{I}) \mathbf{K}_{\mathcal{X}} + 2\beta \mathbf{A} \;=\; 0 \,. \tag{8.14}$$

Since $\mathbf{K}_{\mathcal{X}}$ and $\mathbf{K}_{\mathcal{Y}}$ are symmetric positive semi-definite $m \times m$ matrices, their eigenvalues are all non-negative and their eigenvectors constitute an orthonormal basis of $\mathbb{R}^m$. Thus, $\{u_i v_j^\top\}_{(i,j) \in \mathcal{I}}$ is an orthonormal basis of $\mathbb{R}^{m^2}$. Consequently, w.l.o.g., any $m \times m$ matrix $\mathbf{A}$ can be written as $\mathbf{A} = \sum_{i=1}^m \sum_{j=1}^m \gamma_{i,j} u_i v_j^\top$ for some values of $\gamma_{i,j}$. Hence, we have $\|\mathbf{A}\|^2 = \sum_{i=1}^m \sum_{j=1}^m \gamma_{i,j}^2$ and Equation (8.14) becomes $\gamma_{i,j}(\delta_i \lambda_j^2 + m\beta) = \delta_i \lambda_j (u_i^\top v_j)$. When $\beta = 0$, that equation is solved for $\gamma_{i,j} = 0$ when $\delta_i \lambda_j = 0$ and $\gamma_{i,j} = (u_i^\top v_j)/\lambda_j$ when $\delta_i \lambda_j > 0$ (a solution where the ℓ_2 constraint is not active [5]). When $\beta > 0$, that equation is solved for $\gamma_{i,j} = (\delta_i \lambda_j (u_i^\top v_j))/(\delta_i \lambda_j^2 + m\beta)$ and, in that case, we have $\|\mathbf{A}\|^2 = R^2$ with a unique non-zero solution for β. □

Note that the eigenvectors and eigenvalues of $\mathbf{K}_{\mathcal{X}}$ and $\mathbf{K}_{\mathcal{Y}}$ can be obtained from their singular value decompositions in $O(m^3)$ time. The solution for β can be obtained with Newton's method requiring $\Theta(m^2)$ time for each iteration. Finally, we can obtain $\mathbf{A}$ in $O(m^3)$ by using $\mathbf{A} = \mathbf{u}\boldsymbol{\gamma}\mathbf{v}^\top$ where $\mathbf{u}$ and $\mathbf{v}$ are matrices obtained by concatenating the the column eigenvectors of $\mathbf{K}_{\mathcal{Y}}$ and $\mathbf{K}_{\mathcal{X}}$ respectively and where $\boldsymbol{\gamma}$ denotes the matrix of $\gamma_{i,j}$ values. Hence, the proposed solution of (8.13) is reached in $O(m^3)$ time whenever Newton's method requires at most $O(m)$ iterations.

8.5 Empirical Results

We have compared the solution given by Equation (8.8) (Structured Output by Ridge Regression – SORR) with the one given by Theorem 8 (Structured Output by Sample-Compression–SOSC) on the word recognition task studied by Taskar et al. [2004], Cortes et al. [2007] and the enzyme classification task studied by Rousu et al. [2006]. All hyper-parameters (C, ρ, and kernel parameters) were selected with 10-fold cross-validation (CV) on the training sets where we have relied on the pre-images (using Equation (8.1)) for that purpose only.

The word recognition task consists of predicting the correct word (a sequence of letters) associated to a manuscript picture of the same word. The metrics used for this data set are usually the 0/1-risk (the fraction of errors on words) and the letter risk (the fraction of errors on letters). Hence, following Equation (8.2), we have used the Dirac kernel ($K_{\mathcal{Y}}(y, y') = I(y = y')$) and the Hamming kernel (which is given by the length of the largest string minus the Hamming distance between the two strings). The polynomial kernel of degree d was used for the input kernel. All experiments were done using the protocol described in Taskar et al. [2004]

5. This is the smallest ℓ_2 norm solution of the unconstrained problem. The Moore-Penrose pseudo-inverse of $\mathbf{K}_{\mathcal{X}}$ is also a solution of the unconstrained problem since, in that case, it suffice for $\mathbf{A}$ to satisfy $\mathbf{A}\mathbf{K}_{\mathcal{X}}^2 = \mathbf{K}_{\mathcal{X}}$.

Metric	Dirac kernel		Hamming kernel	
	SORR	**SOSC**	**SORR**	**SOSC**
0/1 risk	0.0518 $\pm$.0078	0.0517 $\pm$.0079	0.0800 $\pm$.0053	0.0799 $\pm$.0053
Letter risk	0.0282 $\pm$.0063	0.0281 $\pm$.0063	0.0374 $\pm$.0051	0.0374 $\pm$.0051

TABLE 8.1 – Empirical results on the word recognition task.

Metric	$\boldsymbol{H-M^3-\ell_\Delta}$	$\boldsymbol{H-M^3-\ell_{\tilde{H}}}$	**SORR**	**SOSC**
0/1 risk	0.957	0.855	0.640	0.684
Confidence intervals	$[0.949, 0.965]$	$[0.840, 0.869]$	$[0.621, 0.659]$	$[0.666, 0.702]$
Hierarchical risk	1.2	2.50	1.71	1.84
F1 Score	0.6330	0.5340	0.5813	0.5569

TABLE 8.2 – Empirical results on the enzyme hierarchical classification task.

and Cortes et al. [2007]. According to Cortes et al. [2007], SORR achieved better performance than structural SVM and max-margin Markov networks. Our empirical results are shown in Table 8.1. The error bars are the standard deviation of the corresponding risk over the different CV folds given by Taskar et al. [2004]. Clearly, SORR and SOSC achieved very similar generalization performance (with overlapping error bars) on both the 0/1-risk and the letter risk.

The enzymes hierarchical classification task consists of predicting a path in an enzyme classification scheme used by biologist to classify amino acid sequences of enzymatic proteins. As in Rousu et al. [2006], the 4-gram kernel was used in the input space. Focussing on the hierarchical risk (the length of the incorrect sub-path from the root to the enzyme leaf) as the most natural metric for this data set, we have used the hierarchical kernel of Jacob et al. [2008] (given by the length of the common sub-path between two paths) on the output space. All experiments were done using the protocol described in Rousu et al. [2006] and our empirical results are shown in Table 8.2. We have also included the results obtained by Rousu et al. [2006] for $H-M^3-\ell_{\tilde{H}}$ and $H-M^3-\ell_\Delta$, which are variants of the max-margin Markov networks. In the case of the 0/1 risk (the fraction of misclassification errors), we have computed the 90% confidence intervals from the binomial tail inversion method of Langford [2005]. From Table 8.2, we see that the 0/1 risk differences between all algorithms are significant (at 0.9 confidence level), with SORR being the best algorithm. For the hierarchical risk, note that from the central limit theorem, the standard deviation of this metric is given by $\sigma/\sqrt{n}$ for a testing set of $n = 1755$ examples when the hierarchical loss variance is σ^2. Since $\sigma \leq 3$ for the hierarchy of 4 levels, the hierarchical risk differences between all algorithms appear to be significant, with $H-M^3-\ell_\Delta$ being the best algorithm.

Metric	Dirac kernel SORR	Dirac kernel SOSC	Hamming kernel SORR	Hamming kernel SOSC
0/1 risk	0.0517 ±.0079	0.0517 ±.0079	0.1328 ±.0128	0.1328 ±.0128
Letter risk	0.0281 ±.0063	0.0281 ±.0063	0.0387 ±.0057	0.0387 ±.0057

TABLE 8.3 – Empirical results on the word recognition task using the quadratic loss for the selection of hyper-parameters.

Metric	SORR	SOSC
0/1 risk	0.6774	0.7202
Confidence intervals	$[0.659, 0.695]$	$[0.702, 0.737]$
Hierarchical risk	1.77	1.87
F1 Score	0.5616	0.5327

TABLE 8.4 – Empirical results on the enzyme hierarchical classification task using the quadratic loss for the selection of hyper-parameters.

8.5.1 Avoiding the pre-image during cross-validation

As stated previously, the proposed learning algorithms do not require solving the pre-image problem during the training phase. To push this idea further, we also tried to avoid solving the pre-image problem for selecting hyper-parameters during cross-validation. In this second round of experiments, we choose hyper-parameters that minimize the quadratic loss of Equation (8.10) instead of the prediction loss. This method is much faster when no efficient algorithm is known to solve the pre-image problem. Empirical results using the quadratic loss for hyper-parameter selection are shown in Table 8.3 and Table 8.4 for the word recognition and the enzymes hierarchical classification task, respectively. We can see that this approach yields competitive but slightly less accurate results for both algorithms on the word recognition task with the Hamming kernel and the enzymes hierarchical classification task. Even though solving the pre-image problem during cross-validation is essential to achieve optimal results, this approach could still be used to speed up the identification of favorable hyper-parameters and thus reduce the search space for optimal hyper-parameters.

8.6 Conclusion

In this chapter, we have shown that the quadratic regression loss is a convex surrogate of the prediction loss when the prediction loss is upper-bounded by the output kernel loss. We have provided two PAC-Bayes upper bounds of the structured prediction risk that depend on the empirical quadratic risk of the deterministic predictor. The second bound, based on the PAC-Bayes sample-compression approach, is more general than the first bound as it holds for

feature spaces that are arbitrary RKHS. The minimizer of the first bound, SORR, turns out to be the predictor proposed by Cortes et al. [2007] while the minimizer of the second bound, SOSC, is a predictor that has been first presented in Giguère et al. [2013b], the paper on which this chapter is based. Both predictors have been compared on practical tasks, yielding similar, but state of the art accuracies. Finally, although it would be time-consuming, it would be interesting to see if we can improve SOSC by using the full set of m^2 constraints instead of the single ℓ_2 constraint used in optimization problem (8.13).

8.7 Appendix

In this appendix, we make use of the following notation. x_i denotes the i^{th} entry of the (column) vector $\boldsymbol{\phi}_{\mathcal{X}}(x)$, y_j the j^{th} entry of the (column) vector $\boldsymbol{\phi}_{\mathcal{Y}}(y)$, $\mathbf{V}[i;j]$ denotes the entry in position (i,j) of the matrix $\mathbf{V}$. Also, $\mathbf{V}[\,;j]$ denotes the j^{th} column of the matrix $\mathbf{V}$. Finally, $\delta_{i,j}$ denotes the delta function which gives 1 if $i=j$, and 0 otherwise.

8.7.1 Example of a distribution where the minimizer of the quadratic risk has a substantial higher error rate than the optimal classifier

We consider a simple one-dimensional binary classification problem where $\mathcal{X} = \mathbb{R}$ and $\mathcal{Y} = \{-1, +1\}$. We thus consider classifiers identified by a single scalar weight w such that the output $h_w(x)$ on an input x is given by $h_w(x) = \text{sgn}(wx)$. Now, consider a distribution D concentrated on four points $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$. Let p_i denote the weight induced by D on x_i. Hence $\sum_{i=1}^4 p_i = 1$. The 0/1 risk is then given by $\sum_{i=1}^4 p_i I(h_w(x_i) \neq y_i)$ and the quadratic risk is given by $\sum_{i=1}^4 p_i (y_i - wx_i)^2$.

Let w_r denote the value of w minimizing the quadratic risk. Since the derivative (with respect to w) of the quadratic risk must vanish at w_r, we find that it is given by the solution of $w_r \sum_{i=1}^4 p_i x_i^2 - \sum_{i=1}^4 p_i y_i x_i = 0$, or equivalently by $w_r = \frac{\sum_{i=1}^4 p_i y_i x_i}{\sum_{i=1}^4 p_i x_i^2}$.

Now let $x_1 = \epsilon$ with $p_1 = (1-\epsilon)/2$ and $y_1 = +1$. Let $x_2 = -\epsilon$ with $p_2 = (1-\epsilon)/2$ and $y_2 = -1$. Let $x_3 = 1/\epsilon$ with $p_3 = \epsilon/2$ and $y_3 = -1$. Let $x_4 = -1/\epsilon$ with $p_4 = \epsilon/2$ and $y_4 = +1$. Then note that, with this distribution, the 0/1 risk of a classifier with a positive weight w is equal to ϵ and the 0/1 risk of a classifier with a negative weight w is equal to $1 - \epsilon$. The difference tends to the maximum value of 1 when ϵ goes to zero.

However, with this distribution, we have $w_r = \frac{-1+\epsilon(1-\epsilon)}{(1-\epsilon)\epsilon^2+(1/\epsilon)}$. Hence w_r is negative for all ϵ between 0 and 1. Hence the 0/1 risk of h_{w_r} is $(1 - \epsilon)$. However there exists classifiers (those with positive w) having a 0/1 risk of ϵ.

8.7.2 Proof of Equation (8.5)

$$\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 \;=\; \|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 + \sigma^2 N_{\mathcal{Y}} \|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 \,. \tag{8.5}$$

Proof. Note that $\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 = \|\boldsymbol{\phi}_{\mathcal{Y}}(y)\|^2 - 2\langle\boldsymbol{\phi}_{\mathcal{Y}}(y), \mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\rangle + \|\mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2$. Let us then compute the expectation according to $Q_{\mathbf{W},\sigma}$ of these three terms.

$$\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \|\boldsymbol{\phi}_{\mathcal{Y}}(y)\|^2 \;=\; \|\boldsymbol{\phi}_{\mathcal{Y}}(y)\|^2$$

$$\begin{aligned}
\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} 2\langle\boldsymbol{\phi}_{\mathcal{Y}}(y), \mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\rangle &= 2\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \langle\boldsymbol{\phi}_{\mathcal{Y}}(y), \sum_{l=1}^{N_{\mathcal{X}}} x_l \mathbf{V}[\,;l]\rangle \\
&= 2\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \sum_{l=1}^{N_{\mathcal{X}}}\sum_{q=1}^{N_{\mathcal{Y}}} y_q \mathbf{V}[q;l]x_l \\
&= 2\sum_{l=1}^{N_{\mathcal{X}}}\sum_{q=1}^{N_{\mathcal{Y}}} y_q x_l \mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \mathbf{V}[q;l] \\
&= 2\sum_{l=1}^{N_{\mathcal{X}}}\sum_{q=1}^{N_{\mathcal{Y}}} y_q x_l \; \mathbf{W}[q;l] \\
&\;\;\vdots \\
&= 2\,\langle\boldsymbol{\phi}_{\mathcal{Y}}(y), \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\rangle
\end{aligned}$$

For $\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \|\mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2$, first note that, since $Q_{\mathbf{W},\sigma}$ is an isotropic Gaussian with mean $\mathbf{W}$ and variance σ^2, we have $\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \mathbf{V}[q;l]\mathbf{V}[q;l] = \mathbf{W}[q;l] + \sigma^2$, and $\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \mathbf{V}[q;l]\mathbf{V}[q;k] = \mathbf{W}[q;l]\mathbf{W}[q;k]$ if $l \neq k$. Thus, we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \|\mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 &= \mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \sum_{l=1}^{N_{\mathcal{X}}}\sum_{k=1}^{N_{\mathcal{X}}} x_l x_k \,\langle \mathbf{V}[\,;l]\,,\, \mathbf{V}[\,;k]\,\rangle \\
&= \sum_{l=1}^{N_{\mathcal{X}}}\sum_{k=1}^{N_{\mathcal{X}}} x_l x_k \sum_{q=1}^{N_{\mathcal{Y}}} \mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \mathbf{V}[q;l]\mathbf{V}[q;k] \\
&= \sum_{l=1}^{N_{\mathcal{X}}}\sum_{\substack{k=1\\k\neq l}}^{N_{\mathcal{X}}} x_l x_k \sum_{q=1}^{N_{\mathcal{Y}}} \mathbf{W}[q;l]\mathbf{W}[q;k] \\
&\qquad + \sum_{k=1}^{N_{\mathcal{X}}} x_k x_k \sum_{q=1}^{N_{\mathcal{Y}}} (\mathbf{W}[q;l]\mathbf{W}[q;k] + \sigma^2) \\
&= \left(\sum_{l=1}^{N_{\mathcal{X}}}\sum_{k=1}^{N_{\mathcal{X}}} x_l x_k \sum_{q=1}^{N_{\mathcal{Y}}} \mathbf{W}[q;l]\mathbf{W}[q;k]\right) + \sum_{k=1}^{N_{\mathcal{X}}} x_k^2 \sum_{q=1}^{N_{\mathcal{Y}}} \sigma^2 \\
&= \|\mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 + \sigma^2 N_{\mathcal{Y}} \|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 \,.
\end{aligned}$$

From all that precedes, we then have

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} \|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 \\
&\qquad = \|\boldsymbol{\phi}_{\mathcal{Y}}(y)\|^2 - 2\langle\boldsymbol{\phi}_{\mathcal{Y}}(y), \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\rangle + \|\mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 + \sigma^2 N_{\mathcal{Y}} \|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2
\end{aligned}$$

$$= \|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 + \sigma^2 N_{\mathcal{Y}} \|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 ,$$

and we are done. □

8.7.3 Proof of Equation (8.6)

Proof. Let us prove that we have

$$\mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} e^{-2\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2} = \left[\frac{\sigma^{N_{\mathcal{X}}}}{\sqrt{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}}\right]^{N_{\mathcal{Y}}} e^{-\frac{2\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}} , \tag{8.6}$$

for the case where each component of vector $\phi_{\mathcal{X}}$ is non-zero. To see that the result will also hold for the case where $\phi_{\mathcal{X}}$ has some zero-valued components, note that the result will hold by replacing $\phi_{\mathcal{X}}$ with $\phi_{\mathcal{X}} + \vec{\epsilon}$, where $\vec{\epsilon}$ is a vector whose entries are all equal to ϵ for an ϵ smaller than the smallest non-zero component of $\phi_{\mathcal{X}}$. The result then comes out from the continuity with respect to $\phi_{\mathcal{X}}$ of the right-hand side of the above equation (taking the limit $\epsilon \to 0$). Now, let

$$I \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\mathbf{V}\sim Q_{\mathbf{W},\sigma}} e^{-2\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2} = \int \frac{d\mathbf{V}}{\left(\sigma\sqrt{2\pi}\right)^{N_{\mathcal{X}}N_{\mathcal{Y}}}} e^{-\frac{1}{2}\frac{\|\mathbf{V}-\mathbf{W}\|^2}{\sigma^2}} e^{-2\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{V}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2} .$$

Performing the change of variables $\mathbf{U} = \mathbf{V} - \mathbf{W}$ gives

$$I = \int \frac{d\mathbf{U}}{\left(\sigma\sqrt{2\pi}\right)^{N_{\mathcal{X}}N_{\mathcal{Y}}}} e^{-\frac{1}{2}\frac{\|\mathbf{U}\|^2}{\sigma^2}} e^{-2\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - (\mathbf{U}+\mathbf{W})\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2} .$$

Now, let $\vec{A}$ be the vector of $\mathcal{H}_{\mathcal{Y}}$ defined as $\vec{A} \stackrel{\text{def}}{=} \boldsymbol{\phi}_{\mathcal{Y}}(y) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)$, and let us denote by A_l, the l^{th} component of the vector $\vec{A}$. Then

$$-2\|\boldsymbol{\phi}_{\mathcal{Y}}(y) - (\mathbf{U}+\mathbf{W})\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 = -2\|\vec{A}\|^2 + -2\|\mathbf{U}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 + 4\langle\vec{A}, \mathbf{U}\boldsymbol{\phi}_{\mathcal{X}}(x)\rangle .$$

This implies that

$$I = e^{-2\|\vec{A}\|^2} \int \frac{d\mathbf{U}}{\left(\sigma\sqrt{2\pi}\right)^{N_{\mathcal{X}}N_{\mathcal{Y}}}} e^{-\frac{1}{2}\left(\frac{\|\mathbf{U}\|^2}{\sigma^2} + 4\|\mathbf{U}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 - 8\langle\vec{A}, \mathbf{U}\boldsymbol{\phi}_{\mathcal{X}}(x)\rangle\right)}. \tag{8.15}$$

An analysis of the argument of the exponential function of I
Let

$$Q \stackrel{\text{def}}{=} \left(\frac{\|\mathbf{U}\|^2}{\sigma^2} + 4\|\mathbf{U}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 - 8\langle\vec{A}, \mathbf{U}\boldsymbol{\phi}_{\mathcal{X}}(x)\rangle\right). \tag{8.16}$$

In the following, A_l denotes the l^{th} component of the vector $\vec{A}$. Then,

$$Q = \sum_{i=1}^{N_{\mathcal{X}}}\sum_{l=1}^{N_{\mathcal{Y}}} \frac{\mathbf{U}_{[l;i]}^2}{\sigma^2} + 4\|\sum_{i=1}^{N_{\mathcal{X}}} \mathbf{U}_{[\,;i]} x_i\|^2 - 8\sum_{i=1}^{N_{\mathcal{X}}} \langle\vec{A}, \mathbf{U}_{[\,;i]}\rangle x_i$$

$$= \sum_{i=1}^{N_{\mathcal{X}}}\sum_{l=1}^{N_{\mathcal{Y}}}\frac{\mathbf{U}_{[l;i]}^2}{\sigma^2} + 4\sum_{i,j=1}^{N_{\mathcal{X}}}\sum_{l=1}^{N_{\mathcal{Y}}}\mathbf{U}_{[l;i]}x_i\mathbf{U}_{[l;j]}x_j - 8\sum_{i,j=1}^{N_{\mathcal{X}}}\sum_{l=1}^{N_{\mathcal{Y}}}\delta_{i,j}A_l\mathbf{U}_{[l;i]}x_i$$

$$= \sum_{i,j=1}^{N_{\mathcal{X}}}\sum_{l=1}^{N_{\mathcal{Y}}}\left(\frac{\delta_{i,j}}{\sigma^2} + 4x_ix_j\right)\mathbf{U}_{[l;i]}\mathbf{U}_{[l;j]} - 8\sum_{i,j=1}^{N_{\mathcal{X}}}\sum_{l=1}^{N_{\mathcal{Y}}}\delta_{i,j}A_l\mathbf{U}_{[l;i]}x_i\,. \tag{8.17}$$

Let us now define the matrices $\mathbf{N}$ and $\mathbf{Z}$, respectively of dimension $N_{\mathcal{X}}\times N_{\mathcal{Y}}$ and $N_{\mathcal{Y}}\times N_{\mathcal{X}}$, as follows, and let us rewrite Equation (8.17) by using them :

$$\mathbf{N}_{[i;j]} \stackrel{\text{def}}{=} \frac{\delta_{i,j}}{\sigma^2} + 4x_ix_j \qquad \text{and} \qquad \mathbf{Z}_{[l;i]} \stackrel{\text{def}}{=} \frac{\mathbf{U}_{[l;i]}}{x_i}\,. \tag{8.18}$$

$$Q = \sum_{l=1}^{N_{\mathcal{Y}}}\left(\sum_{i,j=1}^{N_{\mathcal{X}}}\mathbf{N}_{[i;j]}x_ix_j\mathbf{Z}_{[l;i]}\mathbf{Z}_{[l;j]} - 8\sum_{i=1}^{N_{\mathcal{X}}}A_lx_i^2\mathbf{Z}_{[l;i]}\right)\,. \tag{8.19}$$

The following claim will transform Q in such a way that it will contain a single term including the integration variable $\mathbf{Z}$. This is a consequence of the Fermat's difference of square argument : $(A^2-B^2)=(A-B)(A+B)$.

Claim 1 : For any $l=1,..,N_{\mathcal{Y}}$, let $B_l \stackrel{\text{def}}{=} \frac{4\sigma^2A_l}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}$. Then,

$$Q = \sum_{l=1}^{N_{\mathcal{Y}}}\left(\sum_{i,j=1}^{N_{\mathcal{X}}}\mathbf{N}_{[i;j]}x_ix_j(\mathbf{Z}_{[l;i]}-B_l)(\mathbf{Z}_{[l;j]}-B_l)\right) - \frac{16\|A\|^2\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}\,.$$

Proof of Claim 1. First note that $B_l\left(x_i^2 + 4x_i^2\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2\right) = 4A_lx_i^2\sigma^2$. Then, since $x_i^2=\sum_{j=1}^{N_{\mathcal{X}}}\delta_{i,j}x_ix_j$ and $\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 \stackrel{\text{def}}{=} \sum_{j=1}^{N_{\mathcal{X}}}x_j^2$, we have

$$\sum_{j=1}^{N_{\mathcal{X}}}\mathbf{N}_{[i;j]}x_ix_jB_l = 4A_lx_i^2\,. \tag{8.20}$$

Note also that

$$\begin{aligned}\frac{16\sigma^4A_l^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2} &= B_l^2\left(\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2 + 4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^4\right)\\ &= B_l^2\left(\sum_{i,j=1}^{N_{\mathcal{X}}}\delta_{i,j}x_ix_j + \sum_{i,j=1}^{N_{\mathcal{X}}}4\sigma^2x_i^2x_j^2\right)\\ &= \sum_{i,j=1}^{N_{\mathcal{X}}}\mathbf{N}_{[i;j]}\sigma^2x_ix_jB_l^2\,.\end{aligned}$$

Hence,

$$\sum_{l=1}^{N_{\mathcal{Y}}}\sum_{i,j=1}^{N_{\mathcal{X}}}\left(\mathbf{N}_{[i;j]}x_ix_j(\mathbf{Z}_{[l;i]}-B_l)(\mathbf{Z}_{[l;j]}-B_l)\right) - \frac{16\|A\|^2\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}$$

$$
\begin{aligned}
&= \sum_{l=1}^{N_{\mathcal{Y}}} \left(\sum_{i,j=1}^{N_{\mathcal{X}}} \left(\mathbf{N}_{[i;j]} x_i x_j (\mathbf{Z}_{[l;i]} - B_l)(\mathbf{Z}_{[l;j]} - B_l) \right) - \sum_{i,j=1}^{N_{\mathcal{X}}} \mathbf{N}_{[i;j]} x_i x_j B_l^2 \right) \\
&= \sum_{l=1}^{N_{\mathcal{Y}}} \sum_{i,j=1}^{N_{\mathcal{X}}} \left(\mathbf{N}_{[i;j]} x_i x_j \mathbf{Z}_{[l;i]} \mathbf{Z}_{[l;j]} - \mathbf{N}_{[i;j]} x_i x_j \mathbf{Z}_{[l;i]} B_l - \mathbf{N}_{[i;j]} x_i x_j B_l \mathbf{Z}_{[l;j]} \right. \\
&\qquad \left. + \mathbf{N}_{[i;j]} x_i x_j B_l^2 - \mathbf{N}_{[i;j]} x_i x_j B_l^2 \right) \\
&= \sum_{l=1}^{N_{\mathcal{Y}}} \left(\sum_{i,j=1}^{N_{\mathcal{X}}} \mathbf{N}_{[i;j]} x_i x_j \mathbf{Z}_{[l;i]} \mathbf{Z}_{[l;j]} - 2 \sum_{i=1}^{N_{\mathcal{X}}} \left(\sum_{j=1}^{N_{\mathcal{X}}} \mathbf{N}_{[i;j]} x_i x_j \mathbf{Z}_{[l;i]} B_l \right) \right) \\
&= \sum_{l=1}^{N_{\mathcal{Y}}} \left(\sum_{i,j=1}^{N_{\mathcal{X}}} \mathbf{N}_{[i;j]} x_i x_j \mathbf{Z}_{[l;i]} \mathbf{Z}_{[l;j]} - 2 \sum_{i=1}^{N_{\mathcal{X}}} 4 A_l \mathbf{Z}_{[l;i]} x_i^2 \right) \quad = \quad Q \,.
\end{aligned}
$$

The penultimate equality comes from Equation (8.20). Claim 1 is proved.

Transforming the integral I into a Gaussian integral

Let the operator $\star$: $\{1,..,N_{\mathcal{Y}}\} \times \{1,..,N_{\mathcal{X}}\} \longrightarrow \{1,..,N_{\mathcal{Y}} N_{\mathcal{X}}\}$ be defined as $l \star i \stackrel{\text{def}}{=} (l-1) \cdot N_{\mathcal{X}} + i$. Note that for any $\tilde{l} \in \{1,..,N_{\mathcal{Y}} N_{\mathcal{X}}\}$ there exists a unique 2-tuple $(l,i) \in \{1,..,N_{\mathcal{Y}}\} \times \{1,..,N_{\mathcal{X}}\}$ such that $\tilde{l} = l \star i$.

Let $\vec{z}$ be the vector of dimension $N_{\mathcal{Y}} N_{\mathcal{X}}$ defined as $z_{l \star i} \stackrel{\text{def}}{=} \mathbf{Z}_{[l;i]}$, for any $l \in \{1,..,N_{\mathcal{Y}}\}$, and any $i \in \{1,..,N_{\mathcal{X}}\}$.

Let $\vec{\mu}$ be the vector of dimension $N_{\mathcal{Y}} N_{\mathcal{X}}$ defined as $\mu_{l \star i} \stackrel{\text{def}}{=} B_l$, for any $l \in \{1,..,N_{\mathcal{Y}}\}$, and any $i \in \{1,..,N_{\mathcal{X}}\}$.

Let $\mathbf{M}$ be the matrix of dimension $(N_{\mathcal{Y}} N_{\mathcal{X}}) \times (N_{\mathcal{Y}} N_{\mathcal{X}})$ defined as

$$
\mathbf{M}_{[l \star i \,;\, m \star j]} \stackrel{\text{def}}{=} \delta_{l,m} \mathbf{N}_{[i;j]} x_i x_j \quad \Big(= \delta_{l,m} \big(\tfrac{\delta_{i,j}}{\sigma^2} + 4 x_i x_j x_i x_j \big) \Big), \tag{8.21}
$$

for any $l, m \in \{1,..,N_{\mathcal{Y}}\}$, and any $i, j \in \{1,..,N_{\mathcal{X}}\}$.

In what follows, the reader should interpret $\tilde{l}$ as $l \star i$ and $\tilde{m}$ as $m \star j$. Then,

$$
\begin{aligned}
Q &= \sum_{l=1}^{N_{\mathcal{Y}}} \left(\sum_{i,j=1}^{N_{\mathcal{X}}} \mathbf{N}_{[i;j]} x_i x_j (\mathbf{Z}_{[l;i]} - B_l)(\mathbf{Z}_{[l;j]} - B_l) \right) - \frac{16 \|A\|^2 \sigma^2 \|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1 + 4\sigma^2 \|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2} \\
&= \sum_{m=1}^{N_{\mathcal{Y}}} \left(\sum_{l=1}^{N_{\mathcal{Y}}} \sum_{i,j=1}^{N_{\mathcal{X}}} \left(\delta_{l,m} \mathbf{N}_{[i;j]} x_i x_j (\mathbf{Z}_{[l;i]} - B_l)(\mathbf{Z}_{[l;j]} - B_l) \right) \right) - \frac{16 \|A\|^2 \sigma^2 \|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1 + 4\sigma^2 \|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2} \\
&= \sum_{l=1}^{N_{\mathcal{Y}}} \sum_{i=1}^{N_{\mathcal{X}}} \sum_{m=1}^{N_{\mathcal{Y}}} \sum_{j=1}^{N_{\mathcal{X}}} \left(\delta_{l,m} \mathbf{N}_{[i;j]} x_i x_j (\mathbf{Z}_{[l;i]} - B_l)(\mathbf{Z}_{[l;j]} - B_l) \right) - \frac{16 \|A\|^2 \sigma^2 \|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1 + 4\sigma^2 \|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2} \\
&= \sum_{\tilde{l}=1}^{N_{\mathcal{Y}} N_{\mathcal{X}}} \sum_{\tilde{m}=1}^{N_{\mathcal{Y}} N_{\mathcal{X}}} \Big((z_{\tilde{l}} - \mu_{\tilde{l}}) \; \mathbf{M}_{[\tilde{l};\tilde{m}]} \; (z_{\tilde{m}} - \mu_{\tilde{m}}) \Big) - \frac{16 \|A\|^2 \sigma^2 \|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1 + 4\sigma^2 \|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2} \,.
\end{aligned}
$$

Substituing this expression for Q into the integral I of Equation (8.15) gives

$$I = e^{-2\|\vec{A}\|^2} \int \frac{d\mathbf{U}}{\left(\sigma\sqrt{2\pi}\right)^{N_{\mathcal{X}}N_{\mathcal{Y}}}} e^{-\frac{1}{2}\left(\frac{\|\mathbf{U}\|^2}{\sigma^2}+4\|\mathbf{U}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2-8\langle\vec{A},\mathbf{U}\boldsymbol{\phi}_{\mathcal{X}}(x)\rangle\right)} \tag{8.22}$$

$$= e^{-2\|\vec{A}\|^2} \prod_{i=1}^{N_{\mathcal{X}}} |x_i|^{N_{\mathcal{Y}}} e^{\frac{8\|\vec{A}\|^2\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}} \int \frac{d\vec{z}}{\left(\sigma\sqrt{2\pi}\right)^{N_{\mathcal{X}}N_{\mathcal{Y}}}} e^{-\frac{1}{2}\sum_{\tilde{l}=1}^{N_{\mathcal{Y}}N_{\mathcal{X}}}\sum_{\tilde{m}=1}^{N_{\mathcal{Y}}N_{\mathcal{X}}}\left((z_{\tilde{l}}-\mu_{\tilde{l}})\ \mathbf{M}_{[\tilde{l},\tilde{m}]}\ (z_{\tilde{m}}-\mu_{\tilde{m}})\right)} \tag{8.23}$$

$$= e^{-2\|\vec{A}\|^2} \prod_{i=1}^{N_{\mathcal{X}}} |x_i|^{N_{\mathcal{Y}}} e^{\frac{8\|\vec{A}\|^2\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}} \int \frac{d\vec{z}}{\left(\sigma\sqrt{2\pi}\right)^{N_{\mathcal{X}}N_{\mathcal{Y}}}} e^{-\frac{1}{2}\left((\vec{z}-\vec{\mu})^\top\mathbf{M}(\vec{z}-\vec{\mu})\right)} \tag{8.24}$$

$$= e^{-2\|\vec{A}\|^2} \prod_{i=1}^{N_{\mathcal{X}}} |x_i|^{N_{\mathcal{Y}}} e^{\frac{8\|\vec{A}\|^2\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}} \frac{1}{\sqrt{\det(\mathbf{M})}} \int \frac{d\vec{z}}{\left(\sigma\sqrt{2\pi}\right)^{N_{\mathcal{X}}N_{\mathcal{Y}}}} \frac{1}{\sqrt{\det(\mathbf{M}^{-1})}} e^{-\frac{1}{2}\left((\vec{z}-\vec{\mu})^\top(\mathbf{M}^{-1})^{-1}(\vec{z}-\vec{\mu})\right)} \tag{8.25}$$

$$= e^{-2\|\vec{A}\|^2} \prod_{i=1}^{N_{\mathcal{X}}} |x_i|^{N_{\mathcal{Y}}} e^{\frac{8\|\vec{A}\|^2\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}} \frac{1}{\sqrt{\det(\mathbf{M})}} \cdot 1\,. \tag{8.26}$$

Line (8.24) is a consequence of the fact that $\mathbf{U}_{[l;i]} = x_i\mathbf{Z}_{[l;i]}$ (see Equation (8.18)) and of the fact that $\vec{z}_{\tilde{l}} = \mathbf{Z}_{[l;i]}$. Line (8.26) comes from the fact that the integral of Line (8.25) is an integral of a Gaussian density that, therefore, equals 1. Lines (8.25) and (8.26) force $\mathbf{M}$ to be positive definite.

Claim 2 : The matrix $\mathbf{M}$ *is positive definite and*

$$\det(\mathbf{M}) = \prod_{i=1}^{N_{\mathcal{X}}}(x_i^2)^{N_{\mathcal{Y}}} \left(\frac{1}{\sigma^2}\right)^{N_{\mathcal{X}}N_{\mathcal{Y}}} \left(1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2\right)^{N_{\mathcal{Y}}}\,.$$

Before proving Claim 2, let us show that it implies

$$I = e^{-2\|\vec{A}\|^2} \prod_{i=1}^{N_{\mathcal{X}}} |x_i|^{N_{\mathcal{Y}}} e^{\frac{8\|\vec{A}\|^2\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}} \frac{1}{\sqrt{\det(\mathbf{M})}}$$

$$= e^{-2\|\vec{A}\|^2} \prod_{i=1}^{N_{\mathcal{X}}} |x_i|^{N_{\mathcal{Y}}} e^{\frac{8\|\vec{A}\|^2\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}} \frac{1}{\sqrt{\prod_{i=1}^{N_{\mathcal{X}}}(x_i^2)^{N_{\mathcal{Y}}}\left(\frac{1}{\sigma^2}\right)^{N_{\mathcal{X}}N_{\mathcal{Y}}}(1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2)^{N_{\mathcal{Y}}}}}$$

$$= e^{-2\|\vec{A}\|^2} e^{\frac{8\|\vec{A}\|^2\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}} \frac{1}{\sqrt{\left(\frac{1}{\sigma^2}\right)^{N_{\mathcal{X}}N_{\mathcal{Y}}}(1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2)^{N_{\mathcal{Y}}}}}$$

$$= e^{\frac{-2\|\vec{A}\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}} \frac{\sigma^{N_{\mathcal{X}}N_{\mathcal{Y}}}}{\sqrt{(1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2)^{N_{\mathcal{Y}}}}} = e^{\frac{-2\|\boldsymbol{\phi}_{\mathcal{Y}}(y)-\mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}{1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2}} \frac{\sigma^{N_{\mathcal{X}}N_{\mathcal{Y}}}}{\sqrt{(1+4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2)^{N_{\mathcal{Y}}}}}\,.$$

To finish the proof, let us now prove Claim 2.

Proof of Claim 2. Let $\mathbf{X}$ be the diagonal matrix whose entries are the x_is and note that the matrix $(\mathbf{N}_{[i;j]}x_ix_j)_{i\,;\,j}$ can be expressed as follows :

$$(\mathbf{N}_{[i;j]}x_ix_j)_{i\,;\,j} \;=\; \mathbf{XNX}\,. \tag{8.27}$$

Now, from the definition of $\mathbf{M}$, and basic determinant's properties, we have

$$\begin{aligned}
\det(\mathbf{M}) &= \det\Big((\delta_{l,m}\,\mathbf{N}_{[i;j]}x_ix_j)_{l\star i\,;\,m\star j}\Big) && (8.28)\\
&= \Big(\det\big((\mathbf{N}_{[i;j]}x_ix_j)_{i\,;\,j}\big)\Big)^{N_{\mathcal{Y}}} && (8.29)\\
&= \Big(\det\big(\mathbf{XNX}\big)\Big)^{N_{\mathcal{Y}}} && (8.30)\\
&= \Big(\Big(\textstyle\prod_{i=1}^{N_{\mathcal{X}}} x_i^2\Big)\ \det(\mathbf{N})\Big)^{N_{\mathcal{Y}}}. && (8.31)
\end{aligned}$$

Line (8.28) comes straightforwardly from the definition (see Equation (8.21)). Line (8.29) comes from the fact that $\mathbf{M}$ is a matrix whose entries are all 0, except for $N_{\mathcal{Y}}$ identical blocks of size $N_{\mathcal{X}} \times N_{\mathcal{X}}$ that are positioned in the diagonal of M, each one of these blocks being the matrix $(\mathbf{N}_{[i;j]}x_ix_j)_{i\,;\,j}$. Line (8.31) follows from the fact that $\det(\mathbf{X}) = \Big(\prod_{i=1}^{N_{\mathcal{X}}} x_i\Big)$.

Note also that the block structure of the matrix $\mathbf{M}$ implies that it has exactly the same eigenvalues as Matrix $(\mathbf{N}_{[i;j]}x_ix_j)_{i\,;\,j}$ (but with a multiplicity augmented by a factor of $N_{\mathcal{Y}}$).

Also, it follows from Equation (8.27) that, for each eigenvalue λ of $(\mathbf{N}_{[i;j]}x_ix_j)_{i\,;\,j}$, there exists i such that $\frac{\lambda}{x_i^2}$ is an eigenvalue of $\mathbf{N}$. Indeed, because of Equation (8.27), we have that

$$\det\Big((\mathbf{N}_{[i;j]}x_ix_j)_{i\,;\,j} \;-\; \lambda\mathbf{XX}\Big) = \mathbf{0} \;\;\Leftrightarrow\;\; \det\Big(\mathbf{N} \;-\; \lambda\, I\Big) = \mathbf{0}\,.$$

This, in turn, implies that if $\mathbf{N}$ is positive definite, so is $\mathbf{M}$. Hence, to prove Claim 2, we only have to show that $\mathbf{N}$ is positive definite and $\det(\mathbf{N}) = \big(\frac{1}{\sigma^2}\big)^{N_{\mathcal{X}}}\,\big(1 + 4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2\big)$.

Let us consider a matrix $\mathbf{O}$, defined as $\mathbf{O}_{[i;j]} = 4x_ix_j$. Then, it is easy to see that $\lambda = 0$ is an eigenvalue of $\mathbf{O}$ of multiplicity $N_{\mathcal{X}} - 1$ because the rank of that matrix is 1. Note that line L_i of that matrix is always equal to $\frac{x_i}{x_1}L_1$. Moreover we can easily see that $(x_1, \ldots, x_m)^\top$ is an eigenvector of $\mathbf{O}$ with eigenvalue $4\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2$. Now, note that $\mathbf{N} = \mathbf{O} + \frac{1}{\sigma^2}\cdot I$. Thus, there is a one-to-one correspondence between the eigenvalues of $\mathbf{O}$ and those of $\mathbf{N}$: λ is an eigenvalue of the former if and only if $\lambda + \frac{1}{\sigma^2}$ is an eigenvalue of the latter. Thus N is positive definite, and

$$\det(\mathbf{N}) \;=\; \left(\frac{1}{\sigma^2}\right)^{N_{\mathcal{X}}-1}\left(\frac{1}{\sigma^2} + 4\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2\right) \;=\; \left(\frac{1}{\sigma^2}\right)^{N_{\mathcal{X}}}\big(1 + 4\sigma^2\|\boldsymbol{\phi}_{\mathcal{X}}(x)\|^2\big)\ .$$

□

8.7.4 Proof of $\frac{\partial}{\partial \mathbf{A}} R(\mathbf{A}, S)$ from Theorem (8)

Proof. From Equation (8.9), we have $\mathbf{W} = \sum_{i=1}^{m}\sum_{j=1}^{m} \boldsymbol{\phi}_{\mathcal{Y}}(y_i) A_{[i;j]} \boldsymbol{\phi}^{\dagger}_{\mathcal{X}}(x_j) = \mathbf{M}_{\mathcal{Y}} \mathbf{A} \mathbf{M}^{\dagger}_{\mathcal{X}}$, where $M_{\mathcal{Y}}$ is a $N_{\mathcal{Y}} \times m$ matrix with $\boldsymbol{\phi}_{\mathcal{Y}}(y_i)$ in it's i-th column. Similarly $M_{\mathcal{X}}$ is a $N_{\mathcal{X}} \times m$ matrix with $\boldsymbol{\phi}_{\mathcal{X}}(x_j)$ in it's j-th column.

$$
\begin{aligned}
R(\mathbf{A}, S) &= \frac{1}{m}\sum_{i=1}^{m} \|\boldsymbol{\phi}_{\mathcal{Y}}(y_i) - \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x_i)\|^2 = \frac{1}{m}\|\mathbf{M}_{\mathcal{Y}} - \mathbf{W}\mathbf{M}_{\mathcal{X}}\|^2 \\
&= \frac{1}{m}\|\mathbf{M}_{\mathcal{Y}} - \mathbf{M}_{\mathcal{Y}}\mathbf{A}\mathbf{M}^{\dagger}_{\mathcal{X}}\mathbf{M}_{\mathcal{X}}\|^2 = \frac{1}{m}\|\mathbf{M}_{\mathcal{Y}}(\mathbf{I} - \mathbf{A}\mathbf{K}_{\mathcal{X}})\|^2 .
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial A_{[i;j]}} R(\mathbf{A}, S) &= \frac{1}{m}\frac{\partial}{\partial A_{[i;j]}} \sum_{k,l=1}^{m} [\mathbf{M}_{\mathcal{Y}}(\mathbf{I} - \mathbf{A}\mathbf{K}_{\mathcal{X}})]^2_{[k;l]} \\
&= \frac{2}{m}\sum_{k,l=1}^{m} [\mathbf{M}_{\mathcal{Y}}(\mathbf{I} - \mathbf{A}\mathbf{K}_{\mathcal{X}})]_{[k;l]} \frac{\partial}{\partial A_{[i;j]}} [\mathbf{M}_{\mathcal{Y}}(\mathbf{I} - \mathbf{A}\mathbf{K}_{\mathcal{X}})]_{[k;l]} \\
&= \frac{-2}{m}\sum_{k,l=1}^{m} [\mathbf{M}_{\mathcal{Y}}(\mathbf{I} - \mathbf{A}\mathbf{K}_{\mathcal{X}})]_{[k;l]} \frac{\partial}{\partial A_{[i;j]}} \left[\sum_{k',l'=1}^{m} \mathbf{M}_{\mathcal{Y}_{[k;k']}} \mathbf{A}_{[k';l']} \mathbf{K}_{\mathcal{X}_{[l';l]}} \right] \\
&= \frac{-2}{m}\sum_{k,l=1}^{m} [\mathbf{M}_{\mathcal{Y}}(\mathbf{I} - \mathbf{A}\mathbf{K}_{\mathcal{X}})]_{[k;l]} \mathbf{M}_{\mathcal{Y}_{[k;i]}} \mathbf{K}_{\mathcal{X}_{[j;l]}} \\
&= \frac{-2}{m}\sum_{k,l=1}^{m} (\mathbf{M}_{\mathcal{Y}})^{\dagger}_{[i;k]} [\mathbf{M}_{\mathcal{Y}}(\mathbf{I} - \mathbf{A}\mathbf{K}_{\mathcal{X}})]_{[k;l]} \mathbf{K}_{\mathcal{X}_{[j;l]}} \\
&= \frac{-2}{m}\sum_{l=1}^{m} \left[\mathbf{M}^{\dagger}_{\mathcal{Y}}\mathbf{M}_{\mathcal{Y}}(\mathbf{I} - \mathbf{A}\mathbf{K}_{\mathcal{X}})\right]_{[i;l]} \mathbf{K}_{\mathcal{X}_{[j;l]}} \\
&= \frac{2}{m} [\mathbf{K}_{\mathcal{Y}}(\mathbf{A}\mathbf{K}_{\mathcal{X}} - \mathbf{I})\mathbf{K}_{\mathcal{X}}]_{[i;j]} .
\end{aligned}
$$

□

8.7.5 Details on how Equation (8.14) becomes $\gamma_{i,j}(\delta_i \lambda_j^2 + m\beta) = \delta_i \lambda_j (u_i^{\top} v_j)$

Because $\{u_i v_j^{\top}\}_{(i,j)\in\mathcal{I}}$ constitutes an orthonormal basis of $\mathbb{R}^{m^2}$ we have

$$
\mathbf{A} = \sum_{i=1}^{m}\sum_{j=1}^{m} \gamma_{i,j} u_i v_j^{\top} \tag{8.32}
$$

and since $\mathbf{K}_{\mathcal{Y}} = \sum_{k=1}^{m} \delta_k u_k u_k^{\top}$ and $\mathbf{K}_{\mathcal{X}} = \sum_{l=1}^{m} \lambda_l v_l v_l^{\top}$, we also have

$$
\begin{aligned}
\mathbf{K}_{\mathcal{Y}}\mathbf{K}_{\mathcal{X}} &= \sum_{k=1}^{m} \delta_k u_k u_k^{\top} \sum_{l=1}^{m} \lambda_l v_l v_l^{\top} = \sum_{k,l=1}^{m} \delta_k \lambda_l (u_k^{\top} v_l) u_k v_l^{\top} \\
\mathbf{K}^2_{\mathcal{X}} &= \sum_{l=1}^{m} \lambda_l v_l v_l^{\top} \sum_{l'=1}^{m} \lambda_{l'} v_{l'} v_{l'}^{\top} = \sum_{l=1}^{m} \lambda_l^2 v_l v_l^{\top}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{A}\mathbf{K}_{\mathcal{X}}^2 &= \sum_{k=1}^{m}\sum_{l=1}^{m}\gamma_{k,l}u_k v_l^\top \sum_{l=1}^{m}\lambda_l^2 v_l v_l^\top = \sum_{k=1}^{m}\sum_{l=1}^{m}\gamma_{k,l}\lambda_l^2 u_k v_l^\top \\
\mathbf{K}_{\mathcal{Y}}\mathbf{A}\mathbf{K}_{\mathcal{X}}^2 &= \sum_{k'=1}^{m}\delta_{k'}u_{k'}u_{k'}^\top \sum_{k=1}^{m}\sum_{l=1}^{m}\gamma_{k,l}\lambda_l^2 u_k v_l^\top = \sum_{k=1}^{m}\sum_{l=1}^{m}\gamma_{k,l}\delta_k\lambda_l^2 u_k v_l^\top .
\end{aligned}
$$

Equation (8.14) then becomes

$$
\begin{aligned}
\frac{2}{m}\mathbf{K}_{\mathcal{Y}}(\mathbf{A}\mathbf{K}_{\mathcal{X}}-\mathbf{I})\mathbf{K}_{\mathcal{X}}+2\beta\mathbf{A} &= 0 \\
\frac{2}{m}\mathbf{K}_{\mathcal{Y}}\mathbf{A}\mathbf{K}_{\mathcal{X}}^2-\frac{2}{m}\mathbf{K}_{\mathcal{Y}}\mathbf{K}_{\mathcal{X}}+2\beta\mathbf{A} &= 0 \\
\sum_{k=1}^{m}\sum_{l=1}^{m}\left[\frac{2}{m}\gamma_{k,l}\delta_k\lambda_l^2-\frac{2}{m}\lambda_l(u_k^\top v_l)+2\beta\gamma_{k,l}\right]u_k v_l^\top &= 0 . \qquad (8.33)
\end{aligned}
$$

Since $u_k v_l^\top$ are linearly independent vectors, Equation (8.33) is satisfied when

$$
\begin{aligned}
\frac{2}{m}\gamma_{k,l}\delta_k\lambda_l^2-\frac{2}{m}\lambda_l(u_k^\top v_l)+2\beta\gamma_{k,l} &= 0 \\
\frac{2}{m}\gamma_{k,l}\delta_k\lambda_l^2+2\beta\gamma_{k,l} &= \frac{2}{m}\lambda_l(u_k^\top v_l) \\
\gamma_{k,l}(\delta_k\lambda_l^2+m\beta) &= \delta_k\lambda_l(u_k^\top v_l) .
\end{aligned}
$$

Chapitre 9

Présentation du quatrième article

9.1 Détails de l'article

Titre :	Machine Learning Assisted Design of Highly Active Peptides for Drug Discovery
Auteurs :	Sébastien Giguère, François Laviolette, Mario Marchand, Denise Tremblay, Sylvain Moineau, Éric Biron, Xinxia Liang, Jacques Corbeil
Lieu de publication :	PLOS Computational Biology
Type de publication :	Article journal
Année :	2014

9.2 Contexte

Ces travaux de recherche ont été effectués dans le cadre d'un projet de recherche en équipe subventionné par le Fonds de recherche du Québec - Nature et technologies (FRQNT-Équipe). Le succès de ce projet nous a permis d'obtenir une seconde subvention de $1,5$ M$ octroyée par le Consortium québécois sur la découverte du médicament (CQDM-FOCUS) pour continuer à explorer, entre autres, la méthode présentée ici. Finalement, cet article fut parmi les huit (sur 95 soumissions) à être présentés lors de l'édition 2013 de la rencontre interuniversitaire en pharmacologie organisée par l'Université de Sherbrooke. Le Réseau québécois de recherche sur les médicaments ont offert des prix aux trois meilleures présentations. Le présentateur, Sébastien Giguère, a remporté la seconde place.

9.3 Contributions et discussion

Le nombre de peptides possibles est si grand, qu'il ne sera jamais possible de tous les tester en laboratoire. Pour réduire les coûts et accélérer le processus de découverte, les méthodes d'apprentissage automatique proposent d'apprendre un modèle du phénomène étudié afin de

remplacer partiellement ou en totalité les expérimentations en laboratoire. Cette approche permettra de diminuer les coûts, mais malheureusement, prédire la bioactivité de tous les peptides à l'aide d'un modèle appris prendrait beaucoup trop de temps calcul. Existe-t-il une méthode astucieuse pour identifier les peptides de bioactivité maximale pour certains modèles appris par apprentissage automatique ? Voilà la question qui nous a mené à développer la méthode présentée ici.

Nous proposons un algorithme basé sur les plus longs chemins dans un graphe, similaire au graphe de de Bruijn, pour maximiser la fonction d'inférence des algorithmes à noyaux lorsque le noyau est le *Generic String*. Nous présentons une application de cette méthode en pharmacologie. Un prédicteur de l'activité anti-microbienne de peptides est appris pour prédire les peptides d'activité maximale. Notre hypothèse étant qu'un prédicteur appris à partir de peptides de faible bio-activité peut être utilisé pour identifier des peptides d'activité supérieure.

Les nouveaux peptides identifiés sont synthétisés puis testés en culture contre deux bactéries, *Escherichia coli* et *Staphylococcus aureus*. Ces tests de croissance ont permis de démontrer l'activité anti-microbienne de ces nouveaux peptides. Un des peptides identifiés possède une activité antimicrobienne supérieure à ceux utilisés pour faire l'apprentissage du prédicteur, validant ainsi notre hypothèse.

Nous proposons également une approche assistée par apprentissage automatique à la synthèse combinatoire de peptides. Cette approche permet de calculer certaines statistiques sur la distribution des peptides synthétisés par cette approche combinatoire. En pratique, elle pourrait accélérer la découverte de peptides ayant de hautes bioactivités et diminuer les coûts relatifs à la synthèse et aux expérimentations. La subvention du CQDM nous permettra entre autres d'explorer cette nouvelle approche de synthèse combinatoire assistée par apprentissage automatique.

Notons que le prédicteur d'énergie de liaison entre peptides et protéines présenté au Chapitre 4 pourrait être utilisé avec la méthode présentée ici. Il serait particulièrement intéressant d'apprendre un prédicteur pour une famille de protéines, comme les récepteurs couplés aux protéines G qui sont la cible de 40% des médicaments modernes [Filmore, 2004, Overington et al., 2006]. Ainsi, en apprenant à partir des protéines les mieux étudiés, nous pourrions possiblement trouver des peptides inhibiteurs pour ceux qui le sont moins. La liaison de tels peptides peut se vérifier en laboratoire et le succès d'une telle expérimentation serait un tremplin sans précédent pour les méthodes d'apprentissage automatique comme celles de cette thèse.

Chapitre 10

Machine Learning Assisted Design of Highly Active Peptides for Drug Discovery

The discovery of peptides possessing high biological activity is very challenging due to the enormous diversity for which only a minority have the desired properties. To lower cost and reduce the time to obtain promising peptides, machine learning approaches can greatly assist in the process and even partly replace expensive laboratory experiments by learning a predictor with existing data or with a smaller amount of data generation. Unfortunately, once the model is learned, selecting peptides having the greatest predicted bioactivity often requires a prohibitive amount of computational time. For this combinatorial problem, heuristics and stochastic optimization methods are not guaranteed to find adequate solutions.

We focused on recent advances in kernel methods and machine learning to learn a predictive model with proven success. For this type of model, we propose an efficient algorithm based on graph theory, that is guaranteed to find the peptides for which the model predicts maximal bioactivity. We also present a second algorithm capable of sorting the peptides of maximal bioactivity.

Extensive analyses demonstrate how these algorithms can be part of an iterative combinatorial chemistry procedure to speed up the discovery and the validation of peptide leads. Moreover, the proposed approach does not require the use of known ligands for the target protein since it can leverage recent multi-target machine learning predictors where ligands for similar targets can serve as initial training data. Finally, we validated the proposed approach *in vitro* with the discovery of new cationic anti-microbial peptides.

Source code freely available at `http://graal.ift.ulaval.ca/peptide-design/`.

10.1 Introduction

Drug discovery faces important challenges in terms of cost, complexity and the amount of time required to yield promising compounds. To avoid side effects, a valuable drug precursor must have high affinity with the target protein while minimizing interactions with other proteins. Unfortunately, only a few have such properties and these have to be identified from an astronomical number of candidate compounds. Other factors, such as bioavailability and stability have to be considered ; but this combinatorial search problem, by itself, is very challenging [MEE et al., 1997].

For novel and less studied targets, screening libraries remain the method of choice for rapid data generation. To fully exploit the great conformational and functional diversity, combinatorial peptide chemistry is certainly a powerful tool. A major advantage of using combinatorial peptide libraries over classic combinatorial libraries, where the scaffold is fixed, is the possibility of generating enormous conformational and functional diversity using a randomized synthesis procedure. This chemical diversity and functionality can be further enhanced by the inclusion of non-natural amino acids. Furthermore, having a peptide scaffold can be very informative to screen for similarities in peptidomimetic libraries. For these reasons, this work will focus on using peptides as drug precursors.

However, it is important to note that combinatorial peptide chemistry cannot cover a significant part of the peptide diversity when peptides are longer than a few amino acids. For example, $2g$ of a one-bead one-compound (OBOC) combinatorial library [Lam et al., 1997] composed of randomly-generated peptides of nine residues will generate a maximum of six million compounds, representing a vanishingly small fraction (less than 0.0016%) of the set of all 20^9 peptides. Consequently, it is almost certain that the best peptides will not be present and most synthesized peptides will have low bioactivity. Hence, drug discovery is a combinatorial problem which, unfortunately, cannot be solved using combinatorial chemistry alone. The process of discovering novel compounds with both high bioactivity and low toxicity must therefore be optimized.

Machine learning and kernel methods [Shawe-Taylor and Cristianini, 2004] have the potential to help with this endeavour. These algorithms are extremely effective at providing accurate models for a wide range of biological and chemical problems : anti-cancer activity of small molecules [Swamidass et al., 2005], protein-ligand interactions [Jacob et al., 2008] and protein-protein interactions [Ben-Hur and Noble, 2005]. The inclusion of similarity functions, known as *kernels* [Shawe-Taylor and Cristianini, 2004], provides a novel way to find patterns in biological and chemical data. By incorporating valuable biological and chemical knowledge, kernels provide an efficient way to improve the accuracy of learning algorithms.

This work explores the use of learning algorithms to design and enhance the pharmaceutical properties of compounds [Schneider, 2010, Damborsky and Brezovsky, 2009]. By starting

with a training set containing approximately 100 peptides with their corresponding validated bioactivity (binding affinity, IC_{50}, etc), one can expect that a state-of-the-art kernel method will give a bioactivity model which is sufficiently accurate to find new peptides with activities higher than the 100 used to learn the model. This is possible because each peptide that possesses a small binding affinity contains information about subsequences of residues that can bind to the target. Learning a model can accelerate, but not solve, this costly process. *In-silico* predictions are faster and cheaper than *in-vitro* assays, however, predicting the bioactivity of all possible peptide to select the most bioactive ones would require a prohibitive amount of computational time. Indeed, this transforms the combinatorial drug discovery problem into an equally hard computational task.

We demonstrate that for a large class of kernel based models, it is possible to design an efficient algorithm guaranteed to find the peptide of maximal predicted bioactivity. This algorithm makes use of graph theory and recent work [Giguère et al., 2013] on the prediction of the bioactivity and the binding affinity between peptides and a target protein. This algorithm can be part of an iterative combinatorial chemistry procedure that could speed up the discovery and the validation of peptide leads. Moreover, the proposed approach can be employed without known ligands for the target protein because it can leverage recent multi-target machine learning predictors [Jacob et al., 2008, Giguère et al., 2013] where ligands for similar targets can serve as an initial training set. Finally, we demonstrate the effectiveness and validate our approach *in vitro* by providing an example of how anti-microbial peptides with proven activity were designed.

10.2 Methods

10.2.1 The Generic String kernel

String kernels are symmetric positive semi-definite similarity functions between strings. In our context, strings are sequences of amino acids. Such kernels have been widely used in applications of machine learning to biology. For example, the local-alignment kernel [Saigo et al., 2004], closely related to the well-known Smith-Waterman alignment algorithm, was used for protein homology detection. It was however observed that kernels for large molecules such as proteins were not suitable for smaller amino acid sequences such as peptides [Giguère et al., 2013]. Indeed, the idea of gaps in the local-alignment kernel or in the Smith-Waterman algorithm is well suited for protein homology, but a gap of only a few amino acids in a peptide would have important consequences on its ability to bind with a target protein. Many recently proposed string kernels have emerged from the original idea of the spectrum kernel [Leslie et al., 2002] where each string is represented by the set of all its constituent k-mers. For example, the string $PALI$ can be represented by the set of 2-mers $\{PA, AL, LI\}$. As defined by the k-spectrum kernel, the similarity score between two strings is simply the number of

k-mers that they have in common. For example, the 2-spectrum similarity between $PALI$ and $LIPAT$ is 2, because they have two 2-mers in common (PA and LI).

To characterize the similarity between peptides, two different k-mer criteria were found to be important. First, two k-mers should only contribute to the similarity if they are in similar positions in the two peptides [Meinicke et al., 2004]. Second, the two k-mers should share common physico-chemical properties [Toussaint et al., 2010].

Meinicke and colleagues [Meinicke et al., 2004] proposed to weight the contribution of identical k-mers with a term that decays exponentially with the distance between their positions. If i and j denote the positions of the k-mers in their respective strings, the contribution to the similarity is given by

$$\exp\left(\frac{-(i-j)^2}{2\sigma_p^2}\right), \tag{10.1}$$

where σ_p is a parameter that controls the length of the decay.

Toussaint and colleagues [Toussaint et al., 2010] proposed to consider properties of amino acids when comparing similar k-mers. This was motivated by the fact that amino acids with similar physico-chemical properties can be substituted in a peptide while maintaining the binding characteristics. To capture the physicochemical properties of amino acids, they proposed to use an encoding function $\boldsymbol{\psi} : \mathcal{A} \to \mathbb{R}^d$ where $\boldsymbol{\psi}(a) = (\psi_1(a), \psi_2(a), \ldots, \psi_d(a))$, to map every amino acid $a \in \mathcal{A}$ to a vector where each component $\psi_i(a)$ encodes one of the d properties of amino acid a. In a similar way, we can define $\boldsymbol{\psi}^k : \mathcal{A}^k \to \mathbb{R}^{dk}$ as an encoding function for k-mers, where

$$\boldsymbol{\psi}^k(a_1, a_2, \ldots, a_k) \stackrel{\text{def}}{=} (\boldsymbol{\psi}(a_1), \boldsymbol{\psi}(a_2), \ldots, \boldsymbol{\psi}(a_k)), \tag{10.2}$$

by concatenating k physico-chemical property vectors, each having d components. Throughout this study, the BLOSUM62 matrix was used in such a way that $\boldsymbol{\psi}(a)$ is the line associated to the amino acid a in the matrix. It is now possible to weight the contribution of any two k-mers $a_1, \ldots, a_k$ and $a'_1, \ldots, a'_k$ according to their properties :

$$\exp\left(\frac{-\parallel \boldsymbol{\psi}^k(a_1, \ldots, a_k) - \boldsymbol{\psi}^k(a'_1, \ldots, a'_k) \parallel^2}{2\sigma_c^2}\right), \tag{10.3}$$

where $\|\cdot\|$ denotes the Euclidean distance.

More recently, the Generic String (GS) kernel was proposed for small biological sequences and pseudo-sequences of binding interfaces [Giguère et al., 2013]. The GS kernel similarity between an arbitrary pair $(\mathbf{x}, \mathbf{x}')$ of biological sequences is defined to be

$$GS(\mathbf{x}, \mathbf{x}', k, \sigma_p, \sigma_c) \stackrel{\text{def}}{=} \sum_{l=1}^{k} \sum_{i=0}^{|x|-l} \sum_{j=0}^{|x'|-l} \exp\left(\frac{-(i-j)^2}{2\sigma_p^2}\right) \exp\left(\frac{-\|\boldsymbol{\psi}^l(x_{i+1},..,x_{i+l}) - \boldsymbol{\psi}^l(x'_{j+1},..,x_{j+l})\|^2}{2\sigma_c^2}\right). \tag{10.4}$$

Hence, the GS similarity between strings $\mathbf{x}$ and $\mathbf{x}'$, is given by comparing their 1-mers, 2-mers, ... up to their k-mers, with the position penalizing term of Equation (10.1) and the physico-chemical contribution term of Equation (10.3). The hyper-parameters k, σ_p, σ_c are chosen by cross-validation.

This GS kernel is very versatile since, depending on the chosen hyper-parameters, it can be specialized to eight known kernels [Giguère et al., 2013] : the Hamming kernel, the Dirac delta, the Blended Spectrum [Shawe-Taylor and Cristianini, 2004], the Radial Basis Function (RBF), the Blended Spectrum RBF [Toussaint et al., 2010], the Oligo [Meinicke et al., 2004], the Weighted degree [Rätsch and Sonnenburg, 2004], and the Weighted degree RBF [Toussaint et al., 2010]. It thus follows that the proposed method, based on the GS kernel, is also valid for all of these kernels.

Recently [Giguère et al., 2013], the GS kernel was used to learn a predictor capable of predicting, with reasonable accuracy, the binding affinity of any peptide to any protein on the PepX database. The GS kernel has also outperformed current state-of-the-art methods for predicting peptide-protein binding affinities on single-target and pan-specific Major Histocompatibility Complex (MHC) class II benchmark datasets and three Quantitative Structure Affinity Model benchmark datasets. The GS kernel was also part of a method that won the 2012 Machine Learning Competition in Immunology [Giguère et al., 2013a]. External validation showed that the SVM classifier with the GS kernel was the overall best method to identify, given unpublished experimental data, new peptides naturally processed by the MHC Class I pathway. The proven effectiveness of this kernel made it ideal to tackle the present problem.

10.2.2 The machine learning approach

In the binary classification setting, the learning task is to predict whether a peptide has a specific property such as binding to a target molecule. In the regression setting, the learning task is to predict a real value that quantifies the quality of a peptide, for example, its bioactivity, inhibitory concentration, binding affinity, or bioavailability. In contrast to classification and regression, the task we consider here (described in the next section) is ultimately to predict a string of amino acids.

In this paper, each learning example $((\mathbf{x}, \mathbf{y}), e)$ consists of a peptide $\mathbf{x}$, a drug target $\mathbf{y}$, which is typically a protein (but other biomolecules could be considered), and a real number e representing the bioactivity of the peptide $\mathbf{x}$ with the target $\mathbf{y}$. In classification, $e \in \{+1, -1\}$ denotes whether $(\mathbf{x}, \mathbf{y})$ has the desired property or not. Since predicting real values is strictly more general than predicting binary values, we focused on the more general case of real-valued predictors. Those learning examples are obtained from *in vitro* or *in vivo* experiments. The learning task is therefore to infer the value of e given new examples $(\mathbf{x}, \mathbf{y})$ that would not have been tested through experiments.

A predictor is a function h that returns an output $h(\mathbf{x}, \mathbf{y})$ when given any input $(\mathbf{x}, \mathbf{y})$. In our setting, the output $h(\mathbf{x}, \mathbf{y})$ is a real number that estimates the "true" bioactivity e between $\mathbf{x}$ and $\mathbf{y}$. Such a predictor is said to be multi-target since its output depends on the ligand $\mathbf{x}$ and the target $\mathbf{y}$. A multi-target predictor is generally obtained by learning from numerous peptides, binding to various proteins, for example, a protein family. For this reason, it can predict the bioactivity of any peptide with any protein of the family even if some proteins are not present in the training data.

In contrast, a predictor $h_{\mathbf{y}}(\mathbf{x})$ is said to be *target-specific* when it is dedicated to predict the bioactivity of any peptide $\mathbf{x}$ with a specific protein $\mathbf{y}$. A target-specific predictor is obtained by learning only from peptides binding to a specific protein or from a multi-target predictor. For simplicity, we will focus on target-specific predictor but let us demonstrate how a target-specific predictor is obtained from a multi-target one.

Given a training set $\{((\mathbf{x}_1, \mathbf{y}_1), e_1), \ldots, ((\mathbf{x}_m, \mathbf{y}_m), e_m)\}$, a large class of learning algorithms produce multi-target predictors h with the output $h(\mathbf{x}, \mathbf{y})$ on an arbitrary example $(\mathbf{x}, \mathbf{y})$ given by

$$h(\mathbf{x}, \mathbf{y}) = \sum_{q=1}^{m} \alpha_q k_{\mathcal{Y}}(\mathbf{y}, \mathbf{y}_q) k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}_q)\,, \tag{10.5}$$

where $k_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ and $k_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ are, respectively, the kernel functions between proteins and peptides, and α_q is the weight on the q-th training example. Since we use the GS kernel for $k_{\mathcal{X}}$, we obtain the target-specific predictor

$$h_{\mathbf{y}}(\mathbf{x}) = \sum_{q=1}^{m} \beta_q(\mathbf{y}) GS(\mathbf{x}, \mathbf{x}_q, k, \sigma_p, \sigma_c)\,. \tag{10.6}$$

Here the weight on the q-th training example is now given by $\beta_q(\mathbf{y})$. To obtain $h_{\mathbf{y}}$ from a multi-target predictor, we use $\beta_q(\mathbf{y}) = \alpha_q k_{\mathcal{Y}}(\mathbf{y}, \mathbf{y}_q)$. When $h_{\mathbf{y}}$ is target-specific predictor learned only with peptides binding to $\mathbf{y}$, we simply use $\beta_q(\mathbf{y}) = \alpha_q$. The remainder of this manuscript will focus on target-specific predictor in the form of Equation 10.6. This makes the proposed solution compatible for both target-specific and multi-target predictors.

The weight vector $\boldsymbol{\alpha} \stackrel{\text{def}}{=} (\alpha_1, \ldots, \alpha_m)$ depends on the learning algorithm used, but many algorithms produce prediction functions given by Equation (10.5), including the Support Vector Machine, the Support Vector Regression, the Ridge Regression, and Gaussian Processes. Note that all these learning methods require both kernels to be symmetric and positive semi-definite. This is the case for the GS kernel. The proposed solution for drug design is thus compatible with these popular bioinformatics learning algorithms [Baldi and Brunak, 2001]. However, some machine learning methods such as neural networks and it's derivatives (deep neural networks) are not compatible with the proposed methodology.

Observe that the model of Equation (10.6) is comparable to position specific weighted matrix (PSWM) models when $\beta_q(\mathbf{y}) = 1/m, k = 1, \sigma_p = 0$, and $\sigma_c = 0$. These models are simple and

easily interpretable but have, however, been surpassed by modern machine learning algorithms [Dyrløv Bendtsen et al., 2004].

10.2.3 The combinatorial search problem

The main motivation for learning a predictor from training data is that, once an accurate predictor is obtained, finding druggable peptides would be greatly facilitated. It is true that replacing or reducing the number of expensive laboratory experiments by an *in silico* prediction will reduce costs. However, peptides having a low bioactivity do not qualify as drug precursors. Instead, we should focus on identifying the most bioactive ones. The computational problem is thus to identify and rank peptides according to a specific biological function. Let $\mathcal{A}$ be the set of all amino acids, and $\mathcal{A}^l$ be the set of all possible peptides of length l. Then, finding the peptide $\mathbf{x}^\star \in \mathcal{A}^l$ that, according to $h_{\mathbf{y}}$, has the maximal bioactivity with $\mathbf{y}$, amounts at solving

$$\mathbf{x}^\star_{\mathbf{y}} = \arg\max_{\mathbf{x}\in\mathcal{A}^l} h_{\mathbf{y}}(\mathbf{x})\,. \tag{10.7}$$

This combinatorial problem is complex because, according to the predictor $h_{\mathbf{y}}$, the contribution of an amino acid at a certain position also depend on the $k-1$ adjacent amino acids. This is the case since string kernel use k-mers to compare sequences. For that reason, each amino acids of the peptide cannot be optimized independently, but globally. Moreover, since the number of possible peptides grows exponentially with l (the length of the peptide), a brute force algorithm has an intractable complexity of $\mathcal{O}(|\mathcal{A}|^l \cdot \mathcal{O}(h_{\mathbf{y}}))$ where $\mathcal{O}(h_{\mathbf{y}})$ denotes the worst case time complexity for computing $h_{\mathbf{y}}(\mathbf{x})$, the output of the predictor on peptide a $\mathbf{x}$. Such an algorithm becomes impractical for any peptide exceeding 6 amino acids.

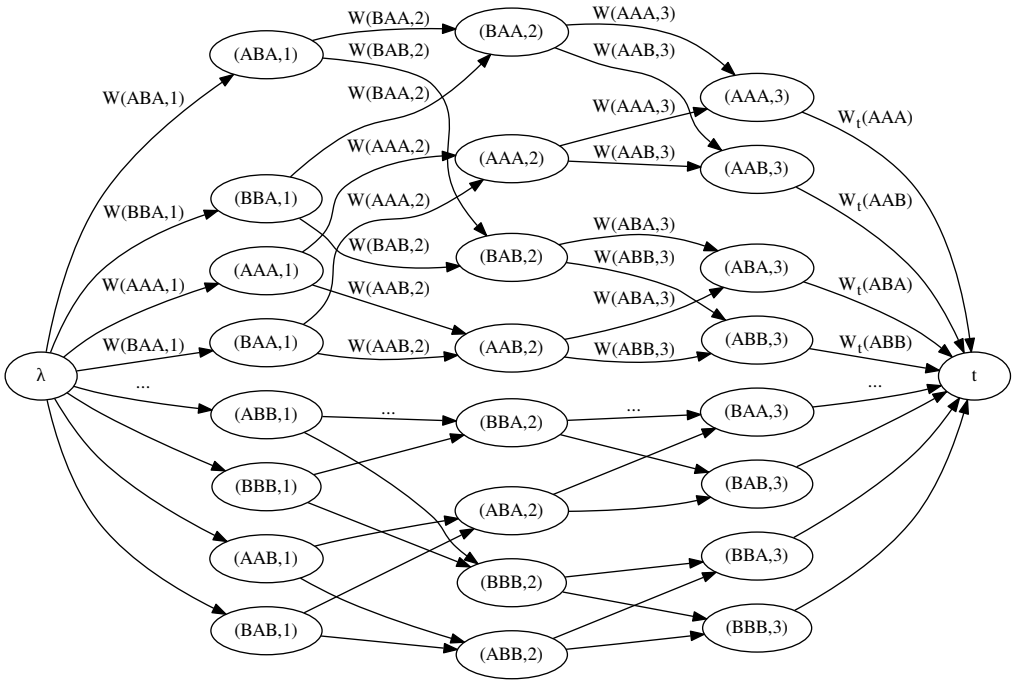
In the next section, we present an efficient algorithm guaranteed to solve Equation (10.7). We also present a second algorithm capable of sorting in decreasing order the peptides maximizing Equation (10.7). Both algorithms have low asymptotic computational complexity, yielding tractable applications for the design and screening of peptides.

When facing such task, heuristics and stochastic optimization methods were generally the methods of choice [Jamois, 2003, Pickett et al., 2000]. However, these methods often require prohibitive CPU time and are not guaranteed to find the optimal solution. In addition, these approaches are not capable of sorting the best solutions since they are designed to find a single maximum.

10.2.4 Finding the peptide of maximal bioactivity

Here, we assume that we have, for a fixed target $\mathbf{y}$, a prediction function $h_{\mathbf{y}}(\mathbf{x})$ given by Equation (10.6). In this case, we show how the problem of finding, the peptide $\mathbf{x}^\star_{\mathbf{y}} \in \mathcal{A}^l$ of maximal bioactivity reduces to the problem of finding the longest path in a directed acyclic graph (DAG). Note that, throughout this manuscript, we will assume that the length of a path

FIGURE 10.1 – Illustration of the 3-partite graph $G^{h_\mathbf{y}}$ with $k = 3$ and a two letters alphabet $\mathcal{A} = \{A, B\}$. In this graph, every source-sink path represent a peptide of size 5 ($l = n + k - 1$) based on the alphabet $\{A, B\}$.



is given by the sum of the weights on its edges. To solve this problem, we construct a DAG with a source and a sink vertex such that for all possible peptides $\mathbf{x} \in \mathcal{A}^l$, there exists only one path associated to $\mathbf{x}$ that goes from the source to the sink. Moreover, the length of the path associated to $\mathbf{x}$ is exactly $h_\mathbf{y}(\mathbf{x})$. Thus, if the size of the constructed graph is polynomial in l, any algorithm that efficiently solves the longest path problem in a DAG will also efficiently find the peptide of maximal bioactivity. A simplification of the graph is shown in Figure 10.1 to assist in the comprehension of the formal definition that follows.

A directed bipartite graph is a graph whose vertices can be divided into two disjoint sets such that every directed edge connects a vertex of the first set to the second set. The construction of the graph will proceed as follows.

Let k be the maximal length of k-mers considered by the GS kernel. Let $U_i \stackrel{\text{def}}{=} \mathcal{A}^k \times \{i\}$, in other words, the set U_i contains all tuples (s, i) where s is a k-mer and i an integer. Let $G_i = ((U_i, U_{i+1}), E_i)$ be the i-th directed bipartite graph of some set where the set of directed edges E_i is defined as follows. Similarly as in the de Bruijn graph, there is a directed edge $((s, i), (s', i + 1))$ from (s, i) in U_i to $(s', i + 1)$ in U_{i+1} if and only if the last $k - 1$ amino acids

of s are the same as the first $k-1$ amino acids of s'. For example, in the graph of Figure 10.1, there is an edge from $(ABA, 1)$ to $(BAA, 2)$ with $k = 3$. Note that $\forall i \in \mathbb{N}$, directed edges in G_i only go from vertices in U_i to vertices in U_{i+1}. There are exactly $|\mathcal{A}|$ edges that leave each vertex in U_i and there are exactly $|\mathcal{A}|$ edges that point to each vertex in U_{i+1}. Moreover, for any chosen integer k, $|U_i| = |U_{i+1}| = |\mathcal{A}^k|$ and $|E_i| = |\mathcal{A}^{k+1}|$. Note that there is a one-to-one correspondence between a sequence in $\mathcal{A}^{k+1}$ and a single edge path from a vertex in U_i to a vertex in U_{i+1}.

We define a n-partite graph as the union of $n-1$ bipartite graphs :

$$G_1 \cup \ldots \cup G_{n-1} \stackrel{\text{def}}{=} ((U_1, U_2, \ldots, U_{n-1}, U_n), E_1 \cup \ldots \cup E_{n-1}) \,.$$

Finally, let $G^{h_{\mathbf{y}}}$ be a n-partite graph with the addition of a source node λ and a sink node t. We choose the letter λ for the source node since it can be interpreted as the empty string (a 0-mer) node. There is a directed edge from λ to all nodes of U_1 and from all nodes of U_n to t. For example, the graph illustrated in Figure 10.1 is a 3-partite graph with a source and a sink node when the k-mer are of size 3 and the alphabet has two letters.

Throughout this manuscript, we will only focus on paths starting at λ, the source node, and ending at t, the sink node. For this reason, by choosing $n = l - k + 1$ we obtain the one-to-one correspondence between each peptide of $\mathcal{A}^l$ and each path $\lambda, u_1, \ldots, u_n, t$ where $u_i \in U_i$. For example, in Figure 10.1 the peptide ABAAA of size $l = 5$ is represented by the path $\lambda, (ABA, 1), (BAA, 2), (AAA, 3), t$.

Let us now describe how edges in $G^{h_{\mathbf{y}}}$ are weighted in order for the length of a path associated to $\mathbf{x}$ to be exactly $h_{\mathbf{y}}(\mathbf{x})$, the predicted bioactivity of $\mathbf{x}$. Using the definition of the GS kernel, given at Equation (10.4), and the general class of predictors, given by Equation (10.6), we can rewrite $h_{\mathbf{y}}(\mathbf{x})$ as

$$h_{\mathbf{y}}(\mathbf{x}) = \sum_{q=1}^{m} \beta_q(\mathbf{y}) \sum_{p=1}^{k} \sum_{i=0}^{|\mathbf{x}|-p} \sum_{j=0}^{|\mathbf{x}_q|-p} \exp\left(\frac{-(i-j)^2}{2\sigma_p^2}\right) \exp\left(\frac{-\|\boldsymbol{\psi}^p(\mathbf{x}_{[i+1]},..,\mathbf{x}_{[i+p]}) - \boldsymbol{\psi}^p(\mathbf{x}_{q[j+1]},..,\mathbf{x}_{q[j+p]})\|^2}{2\sigma_c^2}\right).$$

For any k-mers s and any $i \in \{1, \ldots, n\}$, we define

$$W(s,i) \stackrel{\text{def}}{=} \sum_{q=1}^{m} \beta_q(\mathbf{y}) \sum_{p=1}^{k} \sum_{j=0}^{|\mathbf{x}_q|-p} \exp\left(\frac{-((i-1)-j)^2}{2\sigma_p^2}\right) \exp\left(\frac{-\|\boldsymbol{\psi}^p(s_1,..,s_p) - \boldsymbol{\psi}^p(\mathbf{x}_{q[j+1]},..,\mathbf{x}_{q[j+p]})\|^2}{2\sigma_c^2}\right) \tag{10.8}$$

as the weight on edges heading to the node $(s,i) \in \mathcal{A}^k \times \{1, \ldots, n\}$. The function W weight all edges of $G^{h_{\mathbf{y}}}$ except those heading to the sink vertex t. When $k > 1$, edges $((s,n),t)$, heading to the sink vertex t, are weighted by the function

$$W_t(s) = \sum_{j=1}^{k-1} W(s_{j+1} \ldots s_k, n+j) \,, \tag{10.9}$$

otherwise, $W_t(s) = 0$ when $k = 1$.
For $n = l - k + 1$, we now have that

$$h_{\mathbf{y}}(\mathbf{x}) \;=\; W_t(x_n, .., x_l) + \sum_{i=1}^{n} W(x_i, .., x_{i+k-1}, i)\,.$$

Therefore, every path from the source to the sink in $G^{h_{\mathbf{y}}}$ represents a unique peptide $\mathbf{x} \in \mathcal{A}^l$ and its estimated bioactivity $h_{\mathbf{y}}(\mathbf{x})$ is given by the length of the path.

The problem of finding the peptide of highest predicted activity thus reduces to the problem of finding the longest path in $G^{h_{\mathbf{y}}}$. Despite being NP-hard in the general case, the longest path problem can be solved by dynamic programming in $\mathcal{O}(|V(G)| + |E(G)|)$ for a directed acyclic graph, given a topological ordering of it's vertices. By construction, $G^{h_{\mathbf{y}}}$ is clearly acyclic and its vertices can always be topologically ordered by visiting them in the following order : $\lambda, U_1, \ldots, U_n, t$. Since $G^{h_{\mathbf{y}}}$ has $n|\mathcal{A}|^k + 2$ vertices and $2|\mathcal{A}|^k + (n-1)|\mathcal{A}|^{k+1}$ edges, the complexity of the algorithm will be dominated by the number of edges. Therefore, the proposed algorithm has a complexity of $\mathcal{O}(n|\mathcal{A}|^{k+1})$. Recall that k is a constant and l is the length of the best peptide we are trying to identify. Thus, n must be equal to $l - k + 1$.

Note that Equation (10.8) has to be evaluated for each edges of the graph. The dynamic programming algorithm proposed for the computation of the GS kernel [Giguère et al., 2013] can easily be adapted to efficiently evaluate this equation. In that case, the complexity of the weight function is reduced to $\mathcal{O}(m \cdot l \cdot k)$.

Small values of k are motivated by the fact that $\| \boldsymbol{\psi}^k(a_1, .., a_k) - \boldsymbol{\psi}^k(a'_1, .., a'_k) \|^2$ is a monotonically increasing function of k. Equation (10.3) will thus vanish exponentially fast as k increases. Long k-mers will thus have negligible influence on the estimated bioactivity, explaining why small values of $k \leq 6 \ll l$ are empirically chosen by cross-validation. Therefore, the time complexity of the proposed algorithm is orders of magnitude lower than the brute force algorithm which is in $\mathcal{O}(|\mathcal{A}|^l)$ since $k \leq 6 \ll l$ in practice. The pseudo-code to find the longest path in $G^{h_{\mathbf{y}}}$ is given in Algorithm 1.

10.2.5 Ranking peptides by bioactivity

In the previous section, we demonstrated how the problem of finding the peptide of greatest predicted bioactivity was reduced to the problem of finding a path of maximal length in the graph $G^{h_{\mathbf{y}}}$. By using the same arguments, finding the peptide with the second greatest predicted activity reduces to the problem of finding the second longest path in $G^{h_{\mathbf{y}}}$. By induction, it follows that the problem of finding the K peptides of maximal predicted activity reduces to the problem of finding the K-longest paths in $G^{h_{\mathbf{y}}}$. The closely-related K-shortest paths problem has been studied since 1957 and attracted considerable attention following the work of Yen [1971]. Yen's algorithm was later improved by Lawler [1972]. Both algorithms make use of a shortest path algorithms to solve the K-shortest paths problem. By exploiting some

Algorithm 1 Algorithm for finding the longest path between the source node λ and the sink node t in $G^{h_{\mathbf{y}}}$.

```
length_to = array with n|A|^k + 2 entries initialized to −∞
predecessor = array with n|A|^k + 2 entries

for all s ∈ A^k do                                   ▷ Edges leaving the source node
    length_to[s, 1] ← W(s, 1)
end for

for i = 2 → n do                                     ▷ Edges from the core of G^{h_y}
    for all s ∈ A^k do
        for all a ∈ A do
            s' ← s_2, ..., s_k, a                    ▷ Note that s' is a k-mers
            if length_to[s', i] ≤ length_to[s, i − 1] + W(s', i) then
                length_to[s', i] ← length_to[s, i − 1] + W(s', i)
                predecessor[s', i] ← s
            end if
        end for
    end for
end for

max_length ← −∞
longest_path ← λ
for all s ∈ A^k do                                   ▷ Edges heading to the sink node
    if max_length ≤ length_to[s, n] + W_t(s) then
        max_length ← length_to[s, n] + W_t(s)
        longest_path ← s
    end if
end for

for i = n → 2 do                                     ▷ Backtrack using the predecessors
    s_1, ..., s_k ← predecessor[longest_path_[1:k], i]
    longest_path ← s_1, longest_path
end for

return longest_path
```

restrictive properties of $G^{h_{\mathbf{y}}}$, Yen's algorithm for the K-shortest paths was adapted, shown in Algorithm 2, to find the K-longest paths in $G^{h_{\mathbf{y}}}$. The time complexity of this algorithm is competitive with the latest work on K-shortest paths algorithms [Brander and Sinclair, 1995, Eppstein, 1998]. It uses a variant of the longest path algorithm presented in the previous section, that allows a path to start from any node of the graph.

Algorithm 2 was implemented in a combination of both C and Python, the source code is freely available at `http://graal.ift.ulaval.ca/peptide-design/`. To validate the implementation and prevent potential flaws, it was successfully used to exhaustively rank all peptides of length 1 to 5 with various values of k, σ_p, and σ_c.

Having peptides ranked according to their predicted bioactivity will provide valuable information with the potential of accelerating functional peptide discovery. Indeed, the best peptide candidates can be synthesized quickly by an automated peptide synthesizer and tested *in vitro*. Such a procedure will allow rapid *in vitro* feedback and minimize turnaround time. Also, in the next section, we will describe how the K best predicted peptides can be utilized to predict a binding motif for a new, unstudied protein. Such a motif should assist researchers in the early study of a target and for the design of peptidomimetic compounds by providing residue preferences.

10.2.6 From K-longest paths to motif

It is easy to use the K-longest paths algorithm to predict a motif by simply loading the K peptides to an existing motif tool. In this case, the motif is a property of the learned model $h_{\mathbf{y}}(\mathbf{x})$ as opposed to a consensus among known binding sequences. When $h_{\mathbf{y}}(\mathbf{x})$ is obtained from a multi-target model $h(\mathbf{x}, \mathbf{y})$, it is then possible to predict affinities for proteins with no known ligand by exploiting similarities with related proteins. It is therefore feasible to predict a binding motif for a target with no known binders. To our knowledge, this has never been realized successfully.

10.2.7 Protocol for split and pool peptide synthesis

Split and pool combinatorial peptide synthesis is a simple but efficient way to synthesize a very wide spectrum of peptide ligands. It has been used for the discovery of ligands for receptors [Kumaresan et al., 2011, Liu et al., 2009], for proteins [Alluri et al., 2003, Joo and Pei, 2008, Martínez-Ceron et al., 2011, Zhang et al., 2008] and for transcription factors [Liu et al., 2011, Alluri et al., 2006]. To synthesize several peptides of length l using the 20 natural amino acids, the standard approach is to use one reactor per natural amino acid and a pooling reactor. At every step of the experiment, all reactors are pooled into the pooling reactor which is then split, in equal proportions, back into the 20 amino acid reactors. Within this standard approach, each peptide in $\mathcal{A}^l$ has an equal probability of being synthesized. Since the number of polystyrene beads (used to anchor every peptide) is generally orders of magnitude smaller

Algorithm 2 Algorithm for finding the K-longest paths in $G^{h_{\mathbf{y}}}$

A = array with K entries initialized with the empty string
B = max-heap to store potential paths and their lengths
$A[0] \leftarrow \text{LongestPath}\big(G^{h_{\mathbf{y}}}, \lambda, t\big)$

for $i = 0 \rightarrow K-1$ **do**
  for all $(a, j) \in \big(\lambda, (A[i]_{[0:k]}, 1), .., (A[i]_{[l-k:l]}, n)\big)$ **do** ▷ Nodes of the previous path
    $(V, E) \leftarrow G^{h_{\mathbf{y}}}$
    root $\leftarrow A[i]_{[0:j+k]}$

    for $r = 0 \rightarrow i$ **do**
      if $A[r]_{[0:j+k]} = root$ **then**
        $E \leftarrow E \setminus (A[r]_{[j:j+k]}, j)$
      end if
    end for

    $x \leftarrow$ root + $\text{LongestPath}\big((V, E), (a, j), t\big)$

    if $x \notin B \cup A$ **then**
      $B.push\big(x, h_y(x)\big)$ ▷ Add the string and it's length to the max-heap
    end if
  end for

  $A[i+1] \leftarrow$ B.pop() ▷ B's longest path becomes the i-th longest path
end for

return A

than $|\mathcal{A}|^l$, only a vanishingly small fraction of the peptides in $\mathcal{A}^l$ can be synthesized in each combinatorial experiment.

Clearly, not every peptide has an equal probability of binding to a target. More restrictive protocols have been proposed to increase the hit ratio of this combinatorial experiment. For example, one could fix certain amino acids at specific positions or limit the set of possible amino acids at this position (for example, only use hydrophobic amino acids). Such practice will impact the outcome of the combinatorial experiment. One can probably increase the hit ratio by modifying (wisely) the proportion of amino acids that can be found at different positions in the peptides. To explore more thoroughly this possibility, let us define a (combinatorial chemistry) *protocol* P by a l-tuple containing, for each position i in the peptide of length l, an independent distribution $\mathcal{P}_i(a)$ over the 20 amino acids $a \in \mathcal{A}$. Hence, we define a protocol P by

$$P \stackrel{\text{def}}{=} (\mathcal{P}_1, \ldots, \mathcal{P}_l)\,. \tag{10.10}$$

Consequently, the peptides produced by this protocol will be distributed following the joint distribution $\mathcal{P}_1 \times \ldots \times \mathcal{P}_l$. Hence, the probability of synthesizing a peptide $\mathbf{x}$ of size l is given

by

$$P(\mathbf{x}) = \prod_{i=1}^{l} \mathcal{P}_i(x_i)\,. \tag{10.11}$$

Note that P formally defines a position-specific weight matrix (PSWM) that can be illustrated as a motif. Moreover, this family of protocols is easy to implement in the laboratory since, at each step i, it only requires splitting the content of the pooling reactor in proportions equal to the distribution $\mathcal{P}_i$ over amino acids. For example, if at position i, we wish to sample uniformly over each amino acid, then we will use $\mathcal{P}_i(a) = 1/20$ for all $a \in \mathcal{A}$. If on the other hand, we wish, at position i, to sample amino acids C, D, or E with equal probability and the rest of the amino acids with probability 0, then we use $\mathcal{P}_i(a) = 1/3$ for $a \in \{C, D, E\}$ and $\mathcal{P}_i(a) = 0$ for a different from either C, D, or E.

10.2.8 Expected outcome of a library given a protocol

We present a method for efficiently computing exact statistics on the screening outcome of a peptide library synthesized according to a protocol P. Specifically, we present an algorithm to compute the average predicted bioactivity and its variance over all peptides that a protocol can synthesize. Note that it is intractable to compute these statistics by predicting the activity of each peptide.

Such statistics will, for example, assist chemists in designing a protocol with a greater hit ratio and avoid superfluous experiments. Furthermore, we will demonstrate in the next section that the computation of these statistics can be part of an iterative procedure to accelerate the discovery of bioactive peptides. Indeed, having the average predicted bioactivity data will help with the design of a protocol that synthesizes as many potential active candidates as possible. In addition, the predicted bioactivity variance will allow for better control of the exploration/exploitation trade off of the experiment. Finally, as described in the previous section, a widely used practice for optimizing peptides is to assign residues at certain positions or restrict them to those that have specific properties such as charge or hydrophobicity. It is now possible to quantify how such procedure will impact the bioactivity of combinatorially synthesized peptides.

The proposed approach makes use of the graph $G^{h_\mathbf{y}}$, the protocol P, and a dynamic programming algorithm that exploits recurrences in the factorization of first and second order polynomials. This allows for the efficient computation of τ and β, respectively the first and second moment of $h_\mathbf{y}$ when peptides are drawn according to the distribution P :

$$\begin{aligned} \tau &\stackrel{\text{def}}{=} \sum_{\mathbf{x}\in\mathcal{A}^l} P(\mathbf{x}) \cdot h_\mathbf{y}(\mathbf{x}) \\ \beta &\stackrel{\text{def}}{=} \sum_{\mathbf{x}\in\mathcal{A}^l} P(\mathbf{x}) \cdot h_\mathbf{y}(\mathbf{x})^2\,. \end{aligned}$$

Thus, the average and variance predicted bioactivity of peptides synthesized by the protocol are then respectively given by τ and $\beta - \tau^2$.

To compute these quantities efficiently, Algorithm 3 uses the following recurrence relations :

$$\sum_{i=1}^{n} x_i \quad = \quad x_n + \sum_{i=1}^{n-1} x_i \,, \tag{10.12}$$

and

$$\left(\sum_{i=1}^{n} x_i\right)^2 \quad = \quad \left(\sum_{i=1}^{n-1} x_i\right)^2 + 2x_n \left(\sum_{i=1}^{n-1} x_i\right) + x_n^2 \,. \tag{10.13}$$

Moreover, each node of the graph $G^{h_{\mathbf{y}}}$ has the following additional variables.

— $\boldsymbol{\tau}[s, i]$ for the expected length of paths from the source node λ to the node (s, i).

— $\boldsymbol{\beta}[s, i]$ for the expected squared length of paths from the source node λ to the node (s, i).

— $\boldsymbol{\rho}[s, i]$ is the probability of having the k-mers s at position i.

After the execution of Algorithm 3, the values of τ and β are respectively given by $\boldsymbol{\tau}[t]$ and $\boldsymbol{\beta}[t]$ for the sink node t.

10.2.9 Application in combinatorial drug discovery

We propose an iterative process that makes use of the proposed algorithms to accelerate the discovery of bioactive peptides. The procedure is illustrated in Figure 10.2. First, an initial set of random peptides is synthesized, typically using a split and pool approach. Then, peptides are assayed in laboratory to measure their bioactivities. At this point, most peptides are poor candidates. They are then used as a training set to produce a predictor $h_{\mathbf{y}}$. Next, $h_{\mathbf{y}}$ is used for the generation of K bioactive peptides by finding the K-longest paths in $G^{h_{\mathbf{y}}}$ as described previously. Then, a protocol P is constructed from these K bioactive peptides to assist the next round of combinatorial chemistry. Then, Algorithm 3 is used to predict statistics on the protocol P. This ensures that the protocol meets expectations in terms of quality (average predicted bioactivity) and diversity (predicted bioactivity variance). To lower costs, one should proceed to synthesize and test the library only if expectations are met. This process can be repeated until the desired bioactivity is achieved.

10.3 Results and Discussion

10.3.1 Data

Two public datasets were used to test and validate our approach. The first dataset consisted of 101 cationic antimicrobial pentadecapeptides (CAMPs) from the synthetic antibiotic peptides database [Wade and Englund, 2002]. Peptide antibacterial activities are expressed

Algorithm 3 Algorithm for computing split and pool synthesis statistics using $G^{h_{\mathbf{y}}}$ and $\mathcal{P}$

$\boldsymbol{\tau}, \boldsymbol{\beta}, \boldsymbol{\rho}$: arrays with $n|\mathcal{A}|^k + 2$ entries initialized to 0

for all $s \in \mathcal{A}^k$ **do** ▷ Edges leaving the source node
  $\boldsymbol{\tau}[s, 1] \leftarrow \mathcal{P}_1(s_1)W(s, 1)$
  $\boldsymbol{\beta}[s, 1] \leftarrow \mathcal{P}_1(s_1)W(s, 1)^2$
  $\boldsymbol{\rho}[s, 1] \leftarrow \mathcal{P}_1(s_1)$
end for

for $i = 2 \to n$ **do**
  for all $s \in \mathcal{A}^k$ **do**
    for all $a \in \mathcal{A}$ **do** ▷ Visiting edge $((s, i-1), (s', i))$
      $s' \leftarrow s_2, \ldots, s_k, a$
      $\boldsymbol{\tau}[s', i] += \mathcal{P}_i(s_2)\Big(\boldsymbol{\tau}[s, i-1] + \boldsymbol{\rho}[s, i-1]W(s', i)\Big)$
      $\boldsymbol{\beta}[s', i] += \mathcal{P}_i(s_2)\Big(\boldsymbol{\beta}[s, i-1] + \boldsymbol{\rho}[s, i-1]W(s', i)^2 + 2\boldsymbol{\tau}[s, i-1]W(s', i)\Big)$
      $\boldsymbol{\rho}[s', i] += \mathcal{P}_i(s_2)\boldsymbol{\rho}[s, i-1]$
    end for
  end for
end for

for all $s \in \mathcal{A}^k$ **do** ▷ Edges heading to the sink node
  $r \leftarrow \prod_{i=1}^{k-1} \mathcal{P}_{n+i}(s_{i+1})$
  $\boldsymbol{\tau}[t] += r\Big(\boldsymbol{\tau}[s, n] + \boldsymbol{\rho}[s, n]W_t(s)\Big)$
  $\boldsymbol{\beta}[t] += r\Big(\boldsymbol{\beta}[s, n] + \boldsymbol{\rho}[s, n]W_t((s, n), t)^2 + 2\boldsymbol{\tau}[s, n]W_t(s)\Big)$
  $\boldsymbol{\rho}[t] += r\boldsymbol{\rho}[s, n]$
end for

return $\boldsymbol{\tau}[t],\ \boldsymbol{\beta}[t] - \boldsymbol{\tau}[t]^2$ ▷ Return average and variance

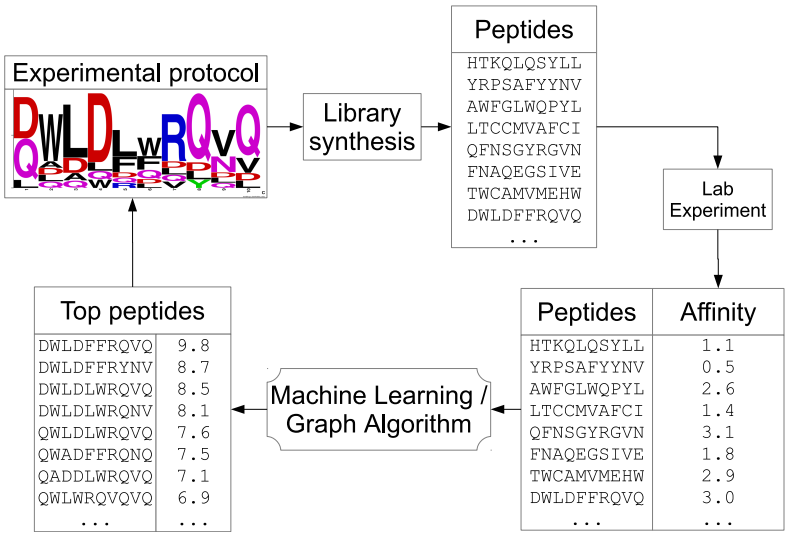
as the logarithm of bactericidal potency which is the average potency over 24 bacteria such as *Escherichia coli, Bacteroïdes fragilis,* and *Staphylococcus aureus*. The average antibacterial activity of the CAMPs dataset was 0.39 and the best peptide had an activity of 0.824.

The second dataset consisted of 31 bradykinin-potentiating pentapeptides (BPPs) reported in Ufkes et al. [1982]. The bioactivities are expressed as the logarithm of the relative activity index compared to the peptide VESSK. The average bioactivity of the BPPs dataset was 0.71 and the best peptide had an activity of 2.73.

10.3.2 Improving the bioactivity of peptides

To assess the capability of the proposed approach to improve upon known peptides, two experiments were carried out using the CAMPs and BPPs peptide datasets. For both experiments, a predictor of biological activity was learned by kernel ridge regression (KRR) for the each

FIGURE 10.2 – Iterative process for the design of peptide ligands.



datasets : h_{CAMP} and h_{BPP}. Hyper-parameters for the GS kernel (k, σ_c, σ_p) and the kernel ridge regression (λ) were chosen by standard cross-validation : $k = 2, \sigma_c = 6.4, \sigma_p = 0.8$, and $\lambda = 6.4$ for h_{CAMP} and $k = 3, \sigma_c = 0.8, \sigma_p = 0.2$, and $\lambda = 0.4$ for h_{BPP}.

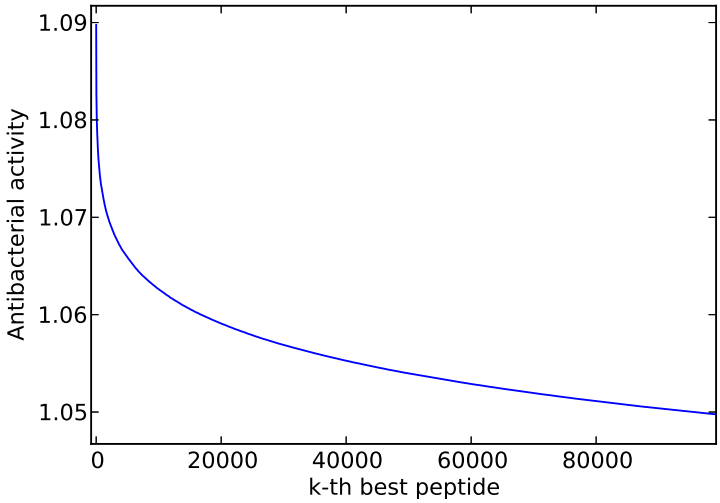
In silico validation

Using the K-longest path algorithm and the learned predictors, we generated the K peptides (of the same length as those of the training data) having the greatest predicted biological activity.

For the CAMPs dataset, the proposed approach predicted that peptide WWKWWKRLRRL-FLLV should have an antibacterial potency of 1.09, a logarithmic improvement of 0.266 over the best peptide in the training set (GWRLIKKILRVFKGL, 0.824), and a substantial improvement over the average potency of that dataset (average of 0.39). The antimicrobial activity of the top $100,000$ peptides are showed in Figure 10.3. We observe a smooth power law with only a few peptides having outstanding biological activity. This is what is expected. As we will see in the next section, peptides at the top of the curve, hence having the best bioactivities, are very unlikely to be found by chance.

On the BPPs dataset, the proposed approach predicted that the pentapeptide IEWAK should have an activity of 2.195, slightly less than the best peptide of the training set (VEWAK, 2.73, predicted as 2.192). However, the predicted activity of IEWAK is much better than the average peptide activity of the dataset, which is 0.71. One may ask why IEWAK has a lower predicted biological activity than VEWAK, which was part of the training data. It is common for machine learning algorithms to sacrifice accuracy on the training data to prevent

FIGURE 10.3 – The 100,000 peptides with highest anti-microbial activity found by the K-longest path algorithm.



overfitting. Despite this small discrepancy, the model is very accurate on the training data (correlation coefficient of 0.97). Another possible explanation for this discrepancy is that the biological activity of VEWAK could be slightly erroneous as the learning algorithm could not find a simple model given such an outlier. It seems that the predicted activity of VEWAK is more coherent with the whole data than its measured activity.

Hence, our proposed learning algorithm is capable to predict new peptides having biological activities equivalent to the best of the training set and, in some cases, substantially improved activities. The next section present an *in vitro* experiment that clearly demonstrate that in a real world test, our approach can generate bioactive peptides.

In vitro validation

To further validate the approach, a number of antimicrobial peptides identified during the *in silico* validation were synthesized. Their antimicrobial activity against *Escherichia coli* and *Staphylococcus aureus* were measured in a growth inhibitory assay. Details on the synthesis and assay are given in the next section. The peptides were obtained using h_{CAMP}, the same predictor used during the previous validation.

The two most active peptides of the CAMPs dataset (Peptide #5 and #6) were synthesized for comparison. We also synthesized one peptide with poor activity (Peptides #7) as a control. We used the proposed approach with the predictor h_{CAMP} to generate a list of $K = 1,000$ peptide candidates with the highest predicted activity. From this list, we greedily selected three peptides such that they all differed by at least 4 amino acids from each others. This was done to maximize the chemical diversity among them. We then tested these peptides

TABLE 10.1 – *In-vitro* minimal inhibitory concentration assay.

#	Predicted Peptide Sequence	MIC ($\mu g/ml$) E. coli	S. aureus	Most Similar Peptide in Training Set Peptide Sequence	% Similarity
1	YWKKWKKLRRIFMLV	2	8	LWKLFKKIRRVLRVL	40.0
2	WWKRWKKLRRIFLML	4	4	LWKLFKKIRRVLRVL	40.0
3	WWKRWKRIRRIFMMV	4	8	LWKLFKKIRRVLRVL	40.0
4	WWKWWKRLRRLFLLV	16	16	LWKLFKKIRRLLKVL	46.6
5	KWKLFKGIRAVLKVL	4	8	-	-
6	GWRLIKKILRVFKGL	4	4	-	-
7	KWKLFLGILAVLKVL	> 32	> 32	-	-

Minimal inhibitory concentration (MIC) from *in vitro* CAMPs assay. We predicted peptides 1 to 4, peptides 5 to 7 are controls from the training set.

(Peptides #2, #3, #4) in a growth inhibitory assay. Results from the minimal inhibitory concentration assay are shown in Table 10.1. Two of the three candidates had activities equal to the best peptide of the CAMPs dataset. We were intrigued by the failure of Peptide #4 and after investigation, the weak activity was due to poor water solubility. In a second series, we ensured that a filter for water solubility was employed. In this second series of tests, Peptide #1 showed (at least against E. coli) better activity than any of the original candidates from the CAMPs dataset, demonstrating that, in this limited biological experiment, we could improve the putative candidates using the proposed machine learning methodology. Finally, all predicted antimicrobial peptides are significantly different from those of the training set, sharing only 40% similarity with their most similar peptide in the CAMPs dataset.

Peptide synthesis, bacterial strains and minimal inhibitory concentration assay

Peptides were synthesized on a Prelude Peptide Synthesizer (Protein Technologies Inc, AZ) using standard Fmoc solid phase peptide chemistry [Wellings and Atherton, 1997]. The synthesis was performed on Rink Amide AM resin and the amino acid couplings achieved with HCTU (2-(6-Chloro-1H-benzotriazole-1-yl)-1,1,3,3-tetramethylaminium hexafluorophosphate) and NMM (N-methylmorpholine). The peptides were cleaved from the resin using a mixture of 95% trifluoroacetic acid, 2.5% triisopropylsilane, 2.5% water for 3h at room temperature and precipitated in cold diethyl ether. After triturating for 2 min, the peptides were collected upon centrifugation and decantation of the ether. The peptides were purified on a Vydac C18 reversed-phase HPLC column (22×250 mm, 5μm) over 20 min using a linear gradient of $10 - 90\%$ acetonitrile with 0.1% trifluoroacetic acid at a flow rate of 10 mL/min with optical density monitoring at 220 nm. The collected fractions were lyophilised and the identity and purity of the peptides assessed by analytical HPLC and MALDI-TOF mass spectrometry. Peptides were obtained in good yields and with purity greater than 90%.

Escherichia coli K12 MG1655 and *Staphylococcus aureus* 68 (HER1049) were obtained from the Félix d'Hérelle Reference Center for Bacterial Viruses of Université Laval (http://www.

phage.ulaval.ca). Both strains were grown in Trypticase soy broth with agitation at $37°C$. The minimal inhibitory concentration assay was performed as described in Wiegand et al. [2008] and broth microdilution protocol performed in 96-well plates. The bacterial strains were grown overnight at $37°C$ with aeration and diluted to a final concentration of 5×10^5 cfu/ml in the assay. The peptides were diluted in sterile water and were tested at the following concentrations : $0, 1, 2, 4, 8, 16$ and 32 μg/ml. The optical density (600nm) was followed every 30 minutes for 24 hours with a Synergy 2 plate reader (BioTek Instruments, Inc.).

10.3.3 Simulation of a drug discovery

Previously, we described a methodology (illustrated in Figure 10.2) that uses machine learning to guide the combinatorial chemistry search for finding peptides with high bioactivity. However, before conducting such an expensive and time-consuming experiment, it is reasonable to first investigate, *in silico*, if the proposed methodology could find peptides having high bioactivity.

Hence, to validate the proposed methodology, we replaced the laboratory experiments that would quantify the bioactivity level of peptides by an oracle for each dataset. We choose to use h_{CAMP} and h_{BPP} as oracle as they represent, so far, the best understanding of the studied phenomena. These oracles will be used to quantify the bioactivity level of randomly generated peptides and those proposed by our methodology. Note that, examples used to learn the oracles are not available to our algorithm during the validation. Consequently, the validation method used was the following.

1. We randomly generated R peptides on a computer instead of using combinatorial chemistry.
2. To measure the bioactivities, we replaced the laboratory experiments by the oracle.
3. We used these random peptides of low bioactivities to learn a second predictor h_{random}.
4. The predictor h_{random} is used to initiate the graph-based approach. We then obtained the K potentially best peptides.
5. The new peptides bioactivities are validated by the oracle (instead of performing laboratory experiments).
6. Finally, we compared the bioactivities of the initial set of peptides (randomly generated) and those proposed by our approach.

Finding peptides with high bioactivity The testing methodology was conducted twice on both the CAMPs and the BPPs datasets. Once by generating $R = 100$ peptides at Step 1 and considering the $K = 100$ best predicted peptides at Step 4 of the methodology, and then by starting over the validation with $R = 1,000$ and $K = 1,000$. Statistics on the random peptides and those proposed by our approach are shown in Table 10.2.

TABLE 10.2 – Results from the drug discovery simulation.

Dataset	Value of R and K	R Randomly Picked Average	R Randomly Picked Max.	K Best Predicted Average	K Best Predicted Max.	h_{random} Correlation Coef.
CAMPs	100	−0.58	0.17	0.76	0.83	0.51
	1000	−0.59	0.18	1.07	1.09	0.90
BPPs	100	0.31	1.39	1.50	2.04	0.67
	1000	0.26	1.36	1.66	2.20	0.93

Bioactivity comparison between the standard combinatorial screening (R random picked peptides) and the proposed approach (K best predicted peptides), initiated with the same R random peptides. Values are logarithm of bactericidal potencies. The correlation coefficients of h_{random} were computed using the oracle.

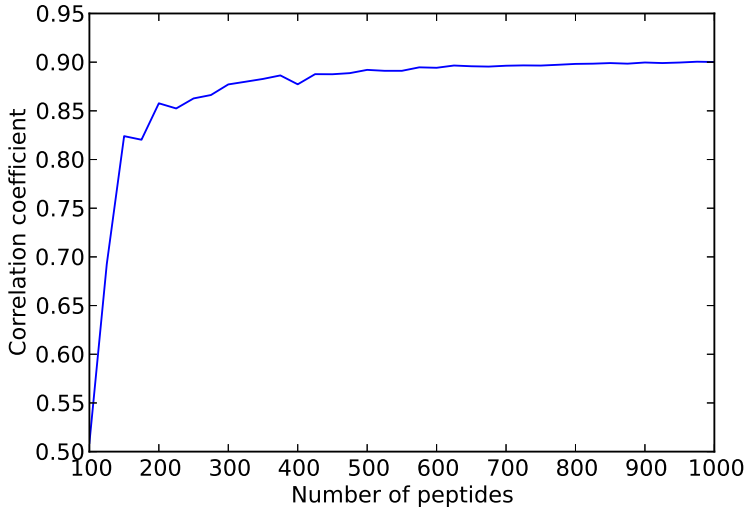
As expected, on both datasets, the number of peptides drawn (R) had no impact on the average activity of randomly drawn peptides. Also, on both datasets, increasing R, the number of random peptides, had no significant influence on the bioactivity of the best peptide found. This support the main hypothesis upon which this work is based, random peptides will consistently be of low activity. This also indicates that combinatorial chemistry alone does not allow one to find the best peptides. It requires hints to orient its search. The next paragraph points out that our machine learning approach can provide such hints.

Using the same $R = 100$ (low bioactivity) random peptides to initiate our method (i.e. train the predictor h_{random}), we were able to reach an antimicrobial potency of 0.83 (according to oracle, not to the prediction of h_{random}). Such antimicrobial potency is similar to the best peptide of the (unseen) CAMPs dataset and much better than the best of the $R = 100$ random peptides. By increasing to R to $1,000$, we found a peptide having potency of 1.09 according to the oracle. This peptide surpasses the best known peptide of the CAMPs dataset and is also far superior to the best of the $R = 1,000$ random peptides. On the BPPs dataset, the proposed approach also considerably outperformed the random approach on both the best peptide found and the average bioactivity. Finally, on both datasets, increasing the number of initial peptides from $R = 100$ to $R = 1,000$ was more beneficial on the bioactivity measures than the random approach.

Comparing h_{random} and the oracle accuracies on the CAMPs and BPPs databases To provide additional support for the accuracy of the predictor h_{random}, it was used to predict bioactivity values of the unseen but *in-vitro* validated peptides of the CAMPs and BPPs databases. The Pearson correlation coefficient (PCC, also known as the Pearson's r) was computed between h_{random} predictions and the values in both databases. Since, in this simulation, h_{random} was learned only with random peptides of low bioactivity, it is interesting to evaluate its accuracy on these databases.

Correlation coefficients are shown in the last column of Table 10.2. When initiated with $R = 1,000$ random peptides, it achieves a correlation coefficient of 0.90 (CAMPs) and 0.93 (BPP).

FIGURE 10.4 – Correlation coefficient of h_{random} predictions on the CAMPs data while varying R, the number of random peptides used as training set.



In comparison, the oracle achieved a correlation coefficient of 0.91 (CAMPs) and 0.97 (BPP) on the same peptides. These were however used to train the oracle. Given that h_{random} is bound to be less accurate than the oracle, these results demonstrate the capability of our approach to learn a predictor using low bioactivity peptides to obtain highly active ones.

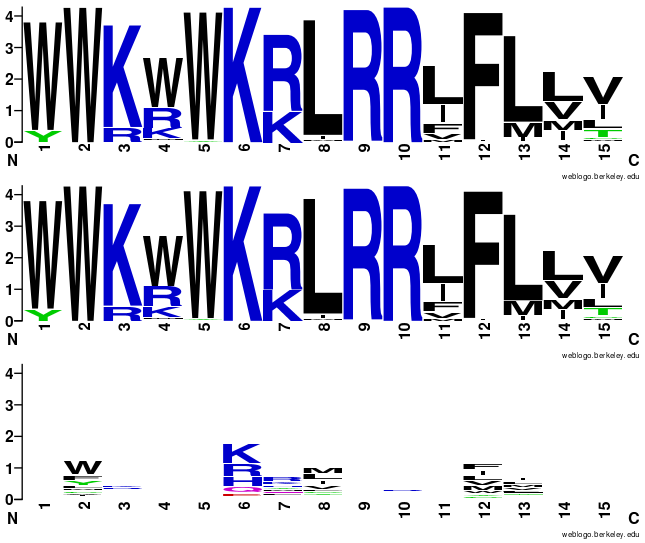
Figure 10.4 shows the correlation coefficient of h_{random} on the CAMPs data when varying R, the number of random peptides used for training. Near optimal accuracy is reached when h_{random} is initiated with approximately $R = 300$ peptides. This suggest that the proposed method can achieve excellent performance with a database of modest size.

Binding motifs results

To demonstrate the ability of the proposed approach to predict potential functional motifs for the CAMPs dataset, we used h_{CAMP} as oracle and hidden all peptides in this dataset from the rest of the procedure. Using the oracle, we predicted the best $K = 1,000$ peptides and generated a bioactivity motif using these candidates (top panel of Figure 10.5). Our goal was to assess how much of that reference motif we could rediscover if we were to hide all the CAMPs dataset to the learning algorithm.

Using the predictor h_{random} that was trained on $R = 1,000$ randomly generated peptides, we generated the motif representing the $K = 1,000$ best predicted peptides. The motif is shown in middle panel of Figure 10.5. We were able to recover all the reference motif signal using only weakly active peptides and h_{random}. This provides evidence that the proposed approach could uncover complex signals for new, poorly understood, proteins. For example, one could learn a

FIGURE 10.5 – CAMP bioactivity motifs. Top motif : the best $1,000$ peptides obtained from the oracle. Middle motif : the best $1,000$ peptides obtained from h_{random}. Bottom motif : the best $1,000$ out of $1,000,000$ random peptides.



multi-target predictor for peptide binding to the major histocompatibility complex [Giguère et al., 2013]. Since these molecules are highly polymorphic, it would be interesting to predict antigen binding motifs for a specific segment of a population or even a single patient. This would have applications in the design of epitope based vaccines [Toussaint and Kohlbacher, 2009] and provide additional insight into autoimmune diseases.

To push the analysis even further, we also computed the motif when h_{random} is trained with only $R = 100$ random peptides. Even then (motif not shown), for 12 residue positions, we were able to correctly identify the dominant amino acid property (polar, neutral, basic, acidic, hydrophobic). This can be achieved since the GS kernel encodes amino acids physico-chemical properties.

To put these results in perspective, we took the same $R = 1,000$ randomly picked peptides used to train the predictor h_{random} and generated a motif from them. The resulting signal was very poor, generating a meaningless motif (not shown). We increased the number of random peptides to $R = 1,000,000$ and only selected the best $K = 1,000$ to produce the motif shown in the bottom panel of Figure 10.5. Even then, the motif shows minimal information. This, again, clearly illustrates the potential of the proposed approach for accelerating the discovery of potential peptidic effectors and possibly better understand the binding of polymorphic molecules.

10.4 Conclusion and Outlook

We proposed an efficient graph-based algorithm to predict peptides with the highest biological activity for machine learning predictors using the GS kernel. Combined with a multi-target model, it can be used to predict binding motifs for targets with no known ligands.

To increase the hit ratio of combinatorial libraries, we demonstrated how a combinatorial chemistry protocol relates to a PSWM. This allowed us to compute the expected predicted bioactivity and its variance that can be exploited in combinatorial chemistry. These steps can be part of an iterative drug discovery process that will have immediate use in both the pharmaceutical industry and academia. This methodology will reduce costs and the time to obtain lead peptides as well as facilitating their optimization. Finally, the proposed approach was validated in a real world test for the discovery of new antimicrobial peptides. These *in vitro* experiments confirmed the effectiveness of the new peptides uncovered.

The K-best peptides were shown to be valuable for the design of split and pool libraries. However, in such libraries, it is unclear how we should prioritize high activity candidates (average) over the chemical diversity (variance). This exploration/exploitation trade-off warrants further investigation. The fast computation of the bioactivity average and variance given a combinatorial chemistry protocol will certainly help to exploit this trade-off. Moreover, the method could easily be adapted to optimize multiple objectives simultaneously, for example, the bioactivity at the expense of mammalian cell toxicity or bioavailability when such data are available. In addition, the method could be expanded to cyclic peptides and chemical entities commonly found in clinical compounds. Finally, this method shows great promise in immunology, where antigen binding motifs for unstudied major histocompatibility complexes could be uncovered using a multi-target predictor.

Acknowledgments

The authors would like to thank Pascal Germain for his insightful comments.

Conclusion

Dans cette thèse, nous avons présenté une série d'articles qui culmine par une nouvelle approche pour la découverte de peptides à valeur pharmaceutique. Les différentes contributions qui nous ont permis d'atteindre cet objectif offrent également des pistes de solutions à plusieurs autres problèmes, par exemple, la conception de vaccins et une meilleure compréhension de la théorie des algorithmes de prédiction de structures. Voici un résumé des contributions de cette thèse, suivi d'une série d'avenues prometteuses à explorer .

Le noyau *generic string*

Nous avons présenté un noyau pour séquences peptidiques qui, lorsqu'utilisé avec des algorithmes d'apprentissage à noyaux, permet d'obtenir des modèles dont la précision surpasse l'état de l'art. Ce noyau généralise huit noyaux déjà proposés. En ce sens, il unifie la théorie derrière ceux-ci. Nous avons fait la preuve que ce noyau satisfait la condition de Mercer, prouvant par le fait même que tous les noyaux généralisés possèdent également cette propriété. Nous avons créé un algorithme par programmation dynamique pour en faire le calcul rapidement. De plus, nous avons proposé un algorithme d'approximation, dont la complexité est linéaire dans la longueur du peptide.

Prédicteur universel d'interactions peptide-protéine

À l'aide de la base de données PepX, qui contient plusieurs complexes peptide-protéine cristallisés à haute définition, nous avons appris un modèle capable de raisonnablement prédire la force d'interaction entre n'importe quelle séquence peptidique et le site de liaison d'une protéine cristallisée. Ce modèle utilise le noyau *generic string* pour les peptides, le noyau Sup-CK pour les protéines cristallisées et la régression de ridge à noyaux comme algorithme d'apprentissage.

Liaison et présentation des antigènes par les MHCs

Le noyau *generic string* s'avère particulièrement adapté pour la prédiction de la liaison des peptides (antigènes) aux complexes majeurs d'histocompatibilité. Celui-ci peut être utilisé autant pour comparer les peptides que les MHCs, en utilisant les résidus polymorphiques de

ceux-ci. Ainsi, nous avons d'abord présenté, au Chapitre 4, des modèles de type *single-target* et *multi-target* pour la prédiction des interactions entre les MHCs et les peptides. Puis, nous avons présenté, au Chapitre 6, un outil permettant de déterminer les antigènes qui seront présentés en surface de la cellule par les MHCs. Cet outil a gagné une compétition internationale pour sa précision.

Bornes PAC-Bayes pour algorithmes de prédiction de structure

Nous proposons deux bornes pour l'approche par régression à la prédiction de structures. Plus spécifiquement, nous montrons que la perte quadratique est un substitut à la perte de prédiction lorsque le noyau de sortie satisfait certaines conditions par rapport à la perte de prédiction. L'utilisation de la perte quadratique permet d'éviter le problème de pré-image, souvent difficile, durant la phase d'apprentissage. Empiriquement, nous démontrons également que le risque quadratique peut raisonnablement substituer la pré-image lors du choix des hyper-paramètres. Les deux bornes proposées bornent supérieurement le risque quadratique de prédiction et suggèrent de minimiser celui-ci ainsi qu'un régularisateur qui varie selon la borne. La première borne suggère un régularisateur L^2 alors que la seconde suggère un régularisateur L^∞. De plus, la première borne est aussi valide pour la régression de ridge à noyau, l'algorithme de régression que nous avons utilisé aux Chapitres 2 et 4 de cette thèse. Finalement, ces bornes apportent une meilleure compréhension des algorithmes de prédiction de structures.

Design de peptides assisté par algorithme d'apprentissage

Nous proposons un algorithme basé sur les K plus longs chemins dans un graph acyclique pour trouver les K peptides de bioactivité maximale d'un modèle appris avec le *generic string*. Nous démontrons l'efficacité de cette approche par la découverte de nouveaux peptides aux propriétés antimicrobiennes. Les champs d'applications de cet algorithme dépassent la biochimie et pourraient avoir des applications pour d'autres problèmes de prédiction de chaînes comme la prédiction de phonèmes ou de mots manuscrits.

Nous proposons également un algorithme pour calculer certaines statistiques sur la distribution des peptides synthétisés par chimie combinatoire. En pratique, cette approche pourrait accélérer la découverte de peptides ayant de hautes bioactivités et diminuer les coûts relatifs à la synthèse et aux expérimentations. Grâce à une série d'expérimentations *in-silico*, nous montrons qu'un modèle appris avec des peptides de faible bioactivité permet l'identification de peptides de bioactivité significativement supérieure. C'est possible parce que chaque peptide de faible bioactivité contient de l'information sur les résidus et structures qui interagissent avec la cible.

Finalement, nous montrons que les K peptides de bioactivité prédite maximale peuvent servir à créer des motifs de liaison ou d'activité biologique. Ces motifs pourront aider à mieux

comprendre, par exemple, la liaison des peptides à certaines protéines polymorphiques comme le complexe majeur d'histocompatibilité.

Travaux futurs

Nous présenterons ici des avenues de recherche prometteuses ou des projets que nous avons débutés mais qui ne furent pas présentés dans cette thèse.

10.4.1 Peptide cycliques

Les peptides cycliques ont un grand potentiel thérapeutique et ont suscité beaucoup d'intérêt dans la découverte de médicaments. Leur capacité à imiter les structures secondaires des protéines (feuillet β, hélice α, ...) a été démontrée. Avec une rigidité structurelle accrue, les peptides cycliques ont de nombreux avantages par rapport à leurs homologues linéaires :

— Ils ont une plus grande stabilité et résistent mieux à la dégradation protéolytique.

— Leur avantage entropique en font des ligands plus spécifiques.

— Dans certain cas, ils ont une biodisponibilité et une perméabilité cellulaire accrues.

— Comme leur analyse conformationelle est plus précise et fiable, ils peuvent être plus facilement optimisés en inhibiteurs.

Nous croyons que le noyau *generic string* pourrait facilement être adapté aux peptides cycliques. Essentiellement, ces molécules n'ont pas de début et de fin. Pour cette raison, il est plus difficile de les aligner. Une approche simple pourrait être de comparer leurs k-mers, mais nous croyons qu'il est possible de faire mieux.

Nous croyons également que l'approche de design de peptide du Chapitre 10 pourrait être adaptée aux peptides cycliques. Déjà, elle est compatible avec l'approche de comparaison des k-mers.

10.4.2 Découverte de peptides inhibiteurs de la protéine NEF

Nos collaborateurs ont fait un criblage de peptides aléatoires contre la protéine NEF du virus d'immunodéficience humaine (VIH). Plusieurs peptides ont été identifiés comme s'y liant. Ils ont été synthétisés, purifiés et leur constante de dissociation sera mesurée par Interférométrie Optique par Nanopores (NPOI) [Latterich and Corbeil, 2008]. Ces données serviront de point de départ pour initier notre méthode de conception de peptides assistée par apprentissage automatique. Nous proposerons de nouveaux peptides dont l'affinité avec la protéine NEF sera également mesurée. Finalement, nous fournirons une distribution (un motif) pour une nouvelle librairie de peptides qui sera obtenue par chimie combinatoire. Nous pourrons mesurer si cette nouvelle librairie nous permet d'obtenir un plus grand nombre de peptides se liant à NEF que la première librairie aléatoire (utilisée comme ensemble d'apprentissage).

10.4.3 Algorithme pour la pré-image du *generic string*

Nous croyons que l'algorithme permettant de maximiser la fonction objective de la régression de ridge avec le *Generic String* (voir Chapitre 10) peut être utilisé pour résoudre le problème de pré-image du même noyau. Ces deux problèmes étant très proches, il est possible que l'algorithme des plus longs chemins puisse être adapté pour résoudre le problème de pré-image. En effet, ce problème s'exprime comme

$$h(\mathbf{x}) = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmin}} \; K_{\mathcal{Y}}(\mathbf{y},\mathbf{y}) - \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_{ij} K_{\mathcal{Y}}(\mathbf{y}_i,\mathbf{y}) K_{\mathcal{X}}(\mathbf{x}_i,\mathbf{x})\,. \tag{10.14}$$

La solution du Chapitre 10 permet déjà de maximiser le second terme de cette équation, mais malheureusement, le premier terme n'est pas constant, même pour les chaînes de longueur fixe.

Si l'algorithme ne peut être adapté, nous croyons qu'un algorithme de recherche de type *branch and bound* puisse être utilisé. Nous croyons que la structure du graphe permettra le calcul rapide d'une borne à utiliser dans l'algorithme de recherche. La découverte d'un algorithme rapide pour résoudre le problème de pré-image des noyaux de chaînes aurait un impact important pour les applications des algorithmes de prédiction de structures.

Annexe A

A Pseudo-Boolean Set Covering Machine

Titre :	A Pseudo-Boolean Set Covering Machine
Auteurs :	Pascal Germain, Sébastien Giguère, Jean-Francis Roy, Brice Zirakiza, François Laviolette, Claude-Guy Quimper
Lieu de publication :	Principles and Practice of Constraint Programming, Springer
Type de publication :	Article de conférence
Année :	2012

A.1 Abstract

The Set Covering Machine (SCM) is a machine learning algorithm that constructs a conjunction of Boolean functions. This algorithm is motivated by the minimization of a theoretical bound. However, finding the optimal conjunction according to this bound is a combinatorial problem. The SCM approximates the solution using a greedy approach. Even though SCM seems very efficient in practice, it is unknown how it compares to the optimal solution. To answer this question, we present a novel pseudo-Boolean optimization model that encodes the minimization problem. It is the first time a Constraint Programming approach addresses the combinatorial problem related to this machine learning algorithm. Using that model and recent pseudo-Boolean solvers, we empirically show that the greedy approach is surprisingly close to the optimal.

A.2 Introduction

Machine learning Bishop [2006] studies algorithms that "learn" to perform a task by observing examples. In the classification framework, a learning algorithm is executed on a training set which contains examples. Each example is characterized by a description and a label. A learning algorithm's goal is to generalize the information contained in the training set to build a

classifier, i.e. a function that takes as input an example description, and outputs a label prediction. A good learning algorithm produces classifiers of low risk, meaning a low probability of misclassifying a new example that was not used in the learning process.

Among all machine learning theories, *Sample Compression* Floyd and Warmuth [1995] studies classifiers that can be expressed by a subset of the training set. This theory allows to compute bounds on a classifier's risk based on two main quantities : the size of the *compression set* (the number of training examples needed to describe the classifier) and the *empirical risk* (the proportion of misclassified training examples). This suggests that a classifier should realize a tradeoff between its complexity, quantified here by the compression set size, and its accuracy on the training set.

Based on this approach, the *Set Covering Machine* (SCM) is a learning algorithm motivated by a sample compression risk bound Marchand and Shawe-Taylor [2002]. However, instead of finding the optimal value of the bound, the SCM algorithm is a greedy approach that aims to quickly find a good solution near the optimal bound's value.

In this paper, we address the following question : "How far to the optimal is the solution returned by the SCM algorithm ?". To answer this question, one needs to design a learning algorithm that directly minimizes the sample compression bound that inspired the SCM. This task is not a trivial one : unlike many popular machine learning algorithms that rely on the minimization of a convex function (as the famous Support Vector Machine Cortes and Vapnik [1995]), this optimization problem is based on a combinatorial function. Although Hussain et al. Hussain et al. [2004] suggested a (convex) linear program version of the SCM, it remains a heuristic inspired by the bound. The present paper describes how to use Constraint Programming techniques to directly minimize the sample compression bound. More precisely, we design a pseudo-Boolean program that encodes the proper optimization problem, and finally show that the SCM is surprisingly accurate.

A.3 Problem Description

The Binary Classification problem in Machine Learning.

An *example* is a pair $(\mathbf{x}, y)$, where $\mathbf{x}$ is a *description* and y is a *label*. In this paper, we consider binary classification, where the description is a vector of n real-valued attributes (i.e. $\mathbf{x} \in \mathbb{R}^n$) and the label is a Boolean value (i.e. $y \in \{0, 1\}$). We say that a 0-labeled example is a *negative example* and a 1-labeled is a *positive example*.

A *dataset* contains several examples coming from the observation of the same phenomenon. We denote S the *training set* of m examples used to "learn" this phenomenon. As the examples are considered to be independently and identically distributed (iid) following a probability

distribution D on $\mathbb{R}^n \times \{0,1\}$, we have :

$$S \stackrel{\text{def}}{=} \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\} \sim D^m .$$

A *classifier* receives as input the description of an example and predicts a label. Thus, a classifier is a function $h : \mathbb{R}^n \to \{0,1\}$. The *risk* $R(h)$ of a classifier is the probability of misclassifying an example generated by the distribution D, and the *empirical risk* $R_S(h)$ of a classifier is the ratio of errors on its training set.

$$R(h) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I(h(\mathbf{x}) \neq y) \quad \text{and} \quad R_S(h) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{(\mathbf{x},y)\in S} I(h(\mathbf{x}) \neq y) ,$$

where I is the indicator function : $I(a) = 1$ if a is true and $I(a) = 0$ otherwise.

A *learning algorithm* receives as input a training set and outputs a classifier. The challenge of a these algorithms is to generalize the information of the training set to produce a classifier of low risk. Since the data generating distribution D is unknown, a common practice to estimate the risk is to calculate the error ratio on a *testing set* containing examples that were not used in the training process.

Overview of the Sample Compression Theory.

The sample compression theory, first expressed by Floyd et al. Floyd and Warmuth [1995], focuses on classifiers that can be expressed by a subset of the training set.

Consider a classifier obtained by executing a learning algorithm on the training set S containing m examples. The *compression set* $S_{\mathbf{i}}$ refers to examples of the training set that are needed to characterize the classifier.

$$S_{\mathbf{i}} \stackrel{\text{def}}{=} \{(\mathbf{x}_{i_1}, y_{i_1}), (\mathbf{x}_{i_2}, y_{i_2}), \ldots, (\mathbf{x}_{i_n}, y_{i_n})\} \subseteq S \quad \text{with } 1 \leq i_1 < i_2 < \ldots < i_n \leq m .$$

We sometimes use a *message string* μ that contains additional information [1]. The term *compressed classifier* refers to the classifier obtained solely with the compression set $S_{\mathbf{i}}$ and message string μ. Sample compression provides theoretical guarantees on a compressed classifier by upper-bounding its risk. Typically, those bounds suggest that a learning algorithm should favour classifiers of low empirical risk (accuracy) and that are expressed by a few training examples (sparsity). One can advocate for sparse classifiers because they are easy to understand by a human being.

The Set Covering Machine.

Suggested by Marchand and Shawe-Taylor Marchand and Shawe-Taylor [2002], the *Set Covering Machine* (SCM) is a learning algorithm directly motivated by the sample compression

1. See Marchand and Shawe-Taylor [2002] for further details about the *message* concept in sample compression theory.

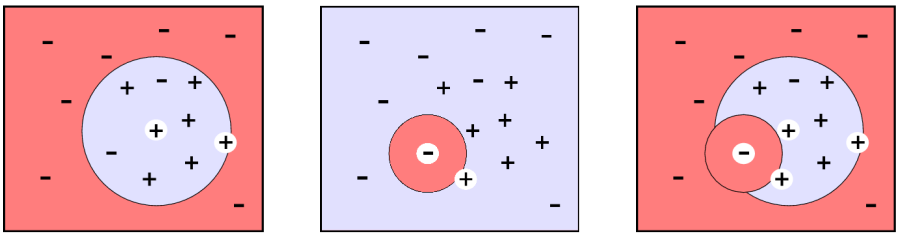


FIGURE A.1 – On a 2-dimensional dataset of 16 examples, from left to right : a positive ball, a negative ball, and the conjunction of both balls. Examples in the light blue region and the red region will be respectively classified positive and negative.

theory. It builds a conjunction or a disjunction of binary functions that rely on training set data. We focus here on the most studied case where each binary function is a *ball* $g_{i,j}$ characterized by two training examples, a center $(\mathbf{x}_i, y_i) \in S$ and a border $(\mathbf{x}_j, y_j) \in S$.

$$g_{i,j}(\mathbf{x}) \stackrel{\text{def}}{=} \begin{cases} y_i \text{ if } \|\mathbf{x}_i - \mathbf{x}\| < \|\mathbf{x}_i - \mathbf{x}_j\| \\ \neg y_i \text{ otherwise,} \end{cases} \tag{A.1}$$

where $\|\cdot\|$ is the Euclidean norm. For simplicity, we omit the case $\|\mathbf{x}_i - \mathbf{x}\| = \|\mathbf{x}_i - \mathbf{x}_j\|$ and consider that a ball correctly classifies its center $(g_{i,j}(\mathbf{x}_i) = y_i)$ and its border $(g_{i,j}(\mathbf{x}_j) = y_j)$.

We denote $\mathcal{H}_S$ the set of all possible balls on a particular dataset S, and $\mathcal{B}$ the set of balls selected by the SCM algorithm among $\mathcal{H}_S$. Thus, the classification function related to a conjunction of balls is expressed by :

$$h_{\mathcal{B}}(\mathbf{x}) \stackrel{\text{def}}{=} \bigwedge_{g \in \mathcal{B}} g(\mathbf{x})\,. \tag{A.2}$$

As the disjunction case is very similar to the conjunction case, we simplify the following discussion by dealing only with the latter [2]. Figure A.1 illustrates an example of a classifier obtained by a conjunction of two balls.

The goal of the SCM algorithm is to choose balls among $\mathcal{H}_S$ to form the conjunction $h_{\mathcal{B}}$. By specializing the sample-compressed classifier's risk bound to the conjunction of balls, Marchand and Sokolova Marchand and Sokolova [2005] proposed to minimize the risk bound given by Theorem 9 below. Note that the compression set $S_{\mathbf{i}}$ contains the examples needed to construct the balls of $h_{\mathcal{B}}$. Also, the message string μ identifies which examples of $S_{\mathbf{i}}$ are centers, and points out the border example associated with each center. In Theorem 9, the variables n_p and n_b encode the length of the message string μ.

Theorem 9. *(Marchand and Sokolova Marchand and Sokolova [2005]) For any data-generating distribution D for which we observe a dataset S of m examples, and for each $\delta \in (0, 1]$:*

$$\Pr_{S \sim D^m} \Big(\forall \mathcal{B} \subseteq \mathcal{H}_S \;:\; R(h_{\mathcal{B}}) \;\leq\; \varepsilon\big(\mathcal{B}\big) \Big) \;\geq\; 1 - \delta\,,$$

2. The disjunction case equations can be recovered by applying De Morgan's law.

where :

$$\varepsilon(\mathcal{B}) \stackrel{def}{=} 1-\exp\left(\frac{-1}{m-(|S_{\mathbf{i}}|+k)}\ln\left[\binom{m}{|S_{\mathbf{i}}|+k}\cdot\binom{|S_{\mathbf{i}}|+k}{k}\cdot\binom{n_p}{n_b}\cdot\frac{1}{\zeta(n_b)\zeta(|S_{\mathbf{i}}|)\zeta(k)\delta}\right]\right), \quad \text{(A.3)}$$

and where k is the number of errors that $h_{\mathcal{B}}$ does on training set S, n_p is the number of positive examples in compression set $S_{\mathbf{i}}$, n_b is the number of different examples used as a border, and $\zeta(a) \stackrel{def}{=} \frac{6}{\pi^2}(a+1)^{-2}$.

This theorem suggests to minimize the expression of $\varepsilon(\mathcal{B})$ in order to find a good balls conjunction. For fixed values of m and δ, this expression tends to decrease with decreasing values of $|S_{\mathbf{i}}|$ and k, whereas $n_b \leq n_p \leq |S_{\mathbf{i}}|$. Moreover, even if the expression of $\varepsilon(\mathcal{B})$ contains many terms, we notice that the quantity $\left[\binom{n_p}{n_b}\cdot\frac{1}{\zeta(n_b)\zeta(|S_{\mathbf{i}}|)\zeta(k)\delta}\right]$ is very small. If we neglect this term, it is easy to see that minimizing $\epsilon(\mathcal{B})$ boils down to find the minimum of Equation (A.4), which is the sum of the compression set size and the number of empirical errors.

$$\mathcal{F}(\mathcal{B}) \stackrel{\text{def}}{=} |S_{\mathbf{i}}| + k. \quad \text{(A.4)}$$

This consideration leads us to the SCM algorithm. We say that a ball belonging to a conjunction *covers an example* whenever it classifies it negatively. Note that a balls conjunction $h_{\mathcal{B}}$ negatively classifies an example $\mathbf{x}$ if and only if at least one ball of $\mathcal{B}$ covers $\mathbf{x}$. This implies that if one wants to add a new ball to an existing balls conjunction, he can only change the classification outcome on uncovered examples. A good strategy for choosing a ball to add to a conjunction is then to cover as few positive examples as possible to avoid misclassifying them. This observation underlies the heuristic of the SCM algorithm.

Given a training set S, the SCM algorithm (see Algorithm 4) is a greedy procedure for selecting a small subset $\mathcal{B}$ of all possible balls[3] so that a high number of negative examples of S are covered by at least one ball belonging to $\mathcal{B}$. At each step of the algorithm, the tradeoff between the number of covered negative examples and the number of covered positive examples is due to a heuristic (Line 7 of Algorithm 4) that depends on a penalty parameter $p \in [0, \infty)$. We initialize the algorithm with a selection of penalty values, allowing it to create a variety of balls conjunctions. The algorithm returns the best conjunction according to a *model selection function* of our choice.

Several model selection functions can be used along with the SCM algorithm. The function ε given by Equation (A.3) leads to excellent empirical results. In other words, by running the algorithm with a variety of penalty parameters, selecting from all generated balls conjunctions the one with the lowest bound value allows to obtain a low risk classifier. This method as been

3. More precisely, the heuristic function (Line 7 of Algorithm 4) makes it possible to consider only balls whose borders are defined by positive examples (see Marchand and Shawe-Taylor [2002]).

Algorithm 4 SCM (*dataset S, penalties $\{p_1, \ldots, p_n\}$, selection function f*)

Consider all possible balls : $\mathcal{H}_S \leftarrow \{g_{i,j} \mid (\mathbf{x}_i, \cdot) \in S, (\mathbf{x}_j, 1) \in S, \mathbf{x}_i \neq \mathbf{x}_j\}$.

Initialize : $\mathcal{B}^* \leftarrow \emptyset$.
for $p \in \{p_1, p_2, \ldots, p_n\}$ **do**
    Initialize : $\mathcal{N} \leftarrow \{\mathbf{x} \mid (\mathbf{x}, 0) \in S\}$, $\mathcal{P} \leftarrow \{\mathbf{x} \mid (\mathbf{x}, 1) \in S\}$ and $\mathcal{B} \leftarrow \emptyset$.
    while $\mathcal{N} \neq \emptyset$ **do**
        Choose the best ball according to the following heuristic :

$$g \leftarrow \underset{g \in \mathcal{H}_S}{\operatorname{argmax}} \left\{ \left| \{\mathbf{x} \in \mathcal{N} \mid g(\mathbf{x}) = 0\} \right| - p \cdot \left| \{\mathbf{x} \in \mathcal{P} \mid g(\mathbf{x}) = 0\} \right| \right\} .$$

        Add this ball to current conjunction : $\mathcal{B} \leftarrow \mathcal{B} \cup \{g\}$.
        Clean covered examples : $\mathcal{N} \leftarrow \{\mathbf{x} \in \mathcal{N} \mid g(\mathbf{x}) = 1\}$, $\mathcal{P} \leftarrow \{\mathbf{x} \in \mathcal{P} \mid g(\mathbf{x}) = 1\}$.
        Retain the best conjunction : **if** $f(\mathcal{B}) < f(\mathcal{B}^*)$ **then** $\mathcal{B}^* \leftarrow \mathcal{B}$.
    end while
end for

shown by Marchand and Shawe-Taylor Marchand and Shawe-Taylor [2002] to be as good as cross-validation [4]. It is exceptional for a risk bound to have such property.

As we explain, the bound relies mainly on the sum $|S_{\mathbf{i}}| + k$, and our extensive experiments with the SCM confirms that the simple model selection function $\mathcal{F}$ given by Equation (A.4) gives equally good results. We are then interested to know if the SCM algorithm provides a good approximation of this function.

To answer this question, next section presents a pseudo-Boolean optimization model that directly finds the set $\mathcal{B}$ that minimizes the function $\mathcal{F}$.

A.4 A pseudo-Boolean optimization model

A pseudo-Boolean problem consists of linear inequality constraints with integer coefficients over binary variables. One can also have a linear objective function.

To solve our machine learning problem with a pseudo-Boolean solver, the principal challenge is to translate the original problem into this particular form. The main strategy to achieve this relies on the following observation :

Observation. As the classification function $h_{\mathcal{B}}$ is a conjunction (see Equation (A.2)), we observe that $h_{\mathcal{B}}$ misclassifies a positive example iff a negative ball covers it. Similarly, $h_{\mathcal{B}}$ misclassifies a negative example iff no ball covers it.

Equivalence rules. Let's first state two general rules that will be useful to express the

4. Cross-validation is a widely used method for estimating reliability of a model, but substantially increases computational needs (see section 1.3 of Bishop [2006]).

problem with pseudo-Boolean constraints. For any positive integer $n \in \mathbb{N}^*$ and Boolean values $\alpha_1, \ldots, \alpha_n, \beta \in \{0, 1\}$, the conjunction and disjunction of the Boolean values α_i can be encoded with these linear inequalities :

$$\alpha_1 \wedge \ldots \wedge \alpha_n = \beta \quad \Leftrightarrow \quad n - 1 \ \geq \ \alpha_1 + \ldots + \alpha_n - n \cdot \beta \ \geq \ 0\,, \tag{A.5}$$

$$\alpha_1 \vee \ldots \vee \alpha_n = \beta \quad \Leftrightarrow \quad 0 \ \geq \ \alpha_1 + \ldots + \alpha_n - n \cdot \beta \ \geq \ 1 - n\,. \tag{A.6}$$

Program variables. Let $P \stackrel{\text{def}}{=} \{i \mid (\mathbf{x}_i, 1) \in S\}$ and $N \stackrel{\text{def}}{=} \{i \mid (\mathbf{x}_i, 0) \in S\}$ be two disjoint sets, containing indices of positive and negative examples respectively. We define m sets B_i, each containing the indices of the borders that can be associated to center $\mathbf{x}_i$, and m sets C_j, each containing the indices of the centers that can be associated to border $\mathbf{x}_j$. As Marchand and Shawe-Taylor Marchand and Shawe-Taylor [2002], we only consider balls with positive borders. Thus, for $i, j \in \{1, \ldots, m\}$, we have :

$$B_i \ \stackrel{\text{def}}{=} \ \{j \mid j \in P,\, j \neq i\} \quad \text{and} \quad C_j \ \stackrel{\text{def}}{=} \ \{i \mid i \in P \cup N,\, j \in B_i\}\ .$$

In other words, B_k is the set of example indices that can be the border of a ball centered on x_k. Similarly, C_k is the set of example indices that can be the center of a ball whose border is x_k. Necessarily, we have $j \in B_k \iff k \in C_j$.

Given the above definitions of B_i and C_j, the solver have to determine the value of Boolean variables s_i, r_i and $b_{i,j}$ described below :

For every $i \in \{1, \ldots, m\}$:

— s_i is equal to 1 iff the example $\mathbf{x}_i$ belongs to the compression set.

— r_i is equal to 1 iff the $h_{\mathcal{B}}$ misclassifies the example $\mathbf{x}_i$.

— For every $j \in B_i$, $b_{i,j}$ is equal to 1 iff the example $\mathbf{x}_i$ is the center of a ball and $\mathbf{x}_j$ if the border of that same ball.

Objective function. The function to optimize (see Equation (A.4)) becomes :

$$\min \sum_{i=1}^{m} (r_i + s_i)\,. \tag{A.7}$$

Program constraints. If an example $\mathbf{x}_i$ is the center of a ball, we want exactly one example $\mathbf{x}_j$ to be its border. Also, if $\mathbf{x}_i$ is not the center of any ball, we don't want any example $\mathbf{x}_j$ to be its border. Those two conditions are encoded by :

$$\sum_{j \in B_i} b_{i,j} \ \leq \ 1 \quad \text{for } i \in \{1, \ldots, m\}\,. \tag{A.8}$$

An example belongs to the compression set iff it is a center or a border. We then have $s_k = \left[\bigvee_{i \in C_k} b_{i,k}\right] \vee \left[\bigvee_{j \in B_k} b_{k,j}\right]$. Equivalence rule (A.6) gives :

$$1 - |B_k \cup C_k| \ \leq \ -|B_k \cup C_k| \cdot s_k + \sum_{i \in C_k} b_{i,k} + \sum_{j \in B_k} b_{k,j} \ \leq \ 0 \ \text{ for } k \in \{1, \ldots, m\}\,. \tag{A.9}$$

We denote by $D_{i,j}$ the distance between examples $\mathbf{x}_i$ and $\mathbf{x}_j$. Therefore, D is a square matrix of size $m \times m$. For each example index $k \in \{1, \ldots, m\}$, let E_k be the set of all balls that cover (i.e. negatively classify) the example $\mathbf{x}_k$:

$$E_k \stackrel{\text{def}}{=} \{b_{i,j} \mid i \in P, j \in B_i, D_{i,j} < D_{i,k}\} \cup \{b_{i,j} \mid i \in N, j \in B_i, D_{i,j} > D_{i,k}\} \ .$$

First, suppose that $\mathbf{x}_k$ is a positive example (thus, $k \in P$). Then, recall that the conjunction misclassifies the example $\mathbf{x}_k$ iff a ball covers it (see "observation" above). Therefore, $r_k = \bigvee_{b_{i,j} \in E_k} b_{i,j}$. Using Equivalence Rule (A.6), we obtain :

$$1 - |E_k| \ \leq \ -|E_k| \cdot r_k + \sum_{b_{i,j} \in E_k} b_{i,j} \ \leq \ 0 \quad \text{for } k \in P \, . \tag{A.10}$$

Now, suppose that $\mathbf{x}_k$ is a negative example (thus, $k \in N$). Then, recall that the conjunction misclassifies $\mathbf{x}_k$ iff no ball covers it (see "observation" above). We have $r_k = \bigwedge_{b_{i,j} \in E_k} \neg\, b_{i,j}$. By using Equivalence Rule (A.5) and $\alpha = \neg\beta \Leftrightarrow \alpha = 1 - \beta$ (where $\alpha, \beta \in \{0, 1\}$), we obtain the following constraints :

$$\begin{aligned} & 0 \ \leq \ -|E_k| \cdot r_k + \sum_{b_{i,j} \in E_k} (1 - b_{i,j}) \ \leq \ |E_k| - 1 \\ \Leftrightarrow \quad & 1 \ \leq \ |E_k| \cdot r_k + \sum_{b_{i,j} \in E_k} b_{i,j} \ \leq \ |E_k| \quad \text{for } k \in N \, . \end{aligned} \tag{A.11}$$

A.5 Empirical Results on Natural Data

The optimization problem of minimizing Equation (A.7) under Constraints (A.8, A.9, A.10, A.11) gives a new learning algorithm that we call *PB-SCM*. To evaluate this new algorithm, we solve several learning problems using three well-known pseudo-Boolean solvers, PWBO Lynce [2011], SCIP Achterberg [2004] and BSOLO Manquinho and Marques-Silva [2006], and compare the obtained results to the SCM (the greedy approach described by Algorithm 4).

We use the same seven datasets than Marchand and Shawe-Taylor [2002] and Marchand and Sokolova [2005], which are common benchmark datasets in the machine learning community. For each dataset, we repeat the following experimental procedure four times with training set sizes $m = |S|$ of 25, 50, 75 and 100 examples. First, we randomly split the dataset examples in a training set S of m examples and a testing set T containing all remaining examples [5]. Then, we execute the four learning algorithms (SCM algorithm and PB-SCM with three different solvers) on the same training set S, and compute the risk on the testing set T.

To obtain SCM results, the algorithm is executed with a set of 41 penalty values $\{10^{a/20} \mid a = 0, 1, \ldots, 40\}$ and the model selection function $\mathcal{F}$ given by Equation (A.4). The PB-SCM problem is solved with the three different solvers. For each solver, we fix the time limit to

5. Training sets are smalls because of the extensive computational power needed by pseudo-Boolean solvers.

TABLE A.1 – Empirical results comparing the objective value $\mathcal{F}$ obtained by SCM and PB-SCM algorithms, the test risk of obtained classifiers and required running time ("T/O" means that the pseudo-Boolean solver reaches the time limit).

Dataset		**SCM**			**PB-SCM** (pwbo)			**PB-SCM** (scip)			**PB-SCM** (bsolo)		
name	**size**	$\mathcal{F}$	**risk**	**time**	$\mathcal{F}$	**risk**	**time**	$\mathcal{F}$	**risk**	**time**	$\mathcal{F}$	**risk**	**time**
breastw	25	**2**	0.046	0.04	**2**	0.081	0.03	**2**	0.064	0.71	**2**	0.046	0.05
	50	**2**	0.047	0.07	**2**	0.046	0.06	**2**	0.049	3.7	**2**	0.047	0.64
	75	**2**	0.044	0.12	**2**	0.041	0.16	**2**	0.044	7.4	**2**	0.044	3.7
	100	**2**	0.046	0.16	**2**	0.046	0.43	**2**	0.05	38	**2**	0.046	20
bupa	25	8	0.403	0.31	**7**	0.45	0.31	**7**	0.45	4.1	**7**	0.419	0.64
	50	14	0.431	1.32	**12**	0.495	589	**12**	0.495	47	**12**	0.464	989
	75	21	0.404	4.1	21	0.463	T/O	**19**	0.467	1763	24	0.419	T/O
	100	**27**	0.355	11	32	0.494	T/O	30	0.396	T/O	34	0.367	T/O
credit	25	**4**	0.202	0.11	**4**	0.202	0.08	**4**	0.202	2	**4**	0.202	0.22
	50	6	0.239	0.25	**5**	0.257	9.3	**5**	0.209	21	**5**	0.257	30.1
	75	9	0.216	0.61	**8**	0.266	1920	**8**	0.263	138	**8**	0.268	1862
	100	12	0.233	1.3	11	0.237	T/O	**10**	0.242	798	18	0.302	T/O
glass	25	**5**	0.333	0.11	**5**	0.261	0.03	**5**	0.297	12	**5**	0.261	0.2
	50	9	0.265	0.49	**8**	0.265	10.3	**8**	0.265	35	**8**	0.265	28
	75	16	0.307	1.5	**15**	0.273	T/O	**15**	0.227	736	**15**	0.227	T/O
	100	18	0.222	2.9	**17**	0.222	T/O	**17**	0.206	T/O	22	0.19	T/O
haberman	25	**5**	0.305	0.17	**5**	0.305	0.03	**5**	0.305	3.6	**5**	0.312	0.18
	50	**10**	0.246	0.94	**10**	0.332	34	**10**	0.332	30	**10**	0.246	65
	75	15	0.237	2.5	**14**	0.324	T/O	**14**	0.324	436	16	0.279	T/O
	100	21	0.278	4.5	**20**	0.289	T/O	**20**	0.33	T/O	23	0.289	T/O
pima	25	**8**	0.408	0.33	**8**	0.381	0.36	**8**	0.385	4	**8**	0.381	0.94
	50	15	0.312	0.9	**13**	0.306	2204	**13**	0.311	37	**13**	0.306	1985
	75	20	0.375	3.8	20	0.342	T/O	**19**	0.339	2641	24	0.336	T/O
	100	25	0.326	7.4	26	0.316	T/O	**23**	0.338	T/O	30	0.379	T/O
USvotes	25	**3**	0.112	0.07	**3**	0.11	0.011	**3**	0.107	0.21	**3**	0.12	0.08
	50	5	0.14	0.17	**4**	0.114	0.141	**4**	0.127	2.4	**4**	0.127	1.1
	75	5	0.119	0.28	**3**	0.131	0.183	**3**	0.131	54	**3**	0.131	33
	100	6	0.084	0.35	**4**	0.146	1.21	**4**	0.107	100	**4**	0.137	80

3600 seconds and keep the solver's default values for other parameters. When a solver fails to converge in 3600 seconds, we consider the best solution so far. Using the solution of the SCM to provide an initial upper bound to the pseudo-Boolean solvers provided no speed-up.

Table A.1 shows the obtained results. Of course, except for T/O situations, the minimal value of the heuristic $\mathcal{F}$ is always obtained by solving the PB-SCM problem. However, it is surprising that the SCM often reaches the same minimum value. Moreover, the SCM sometimes (quickly) finds a best value of $\mathcal{F}$ when the pseudo-Boolean programs time out, and there is no clear amelioration of the testing risk when PB-SCM finds a slightly better solution than SCM. We conclude that the greedy strategy of SCM is particularly effective.

A.6 Conclusion

We have presented a pseudo-Boolean model that encodes the core idea behind the combinatorial problem related to the Set Covering Machine. Extensive experiments have been done using three different pseudo-Boolean solvers. For the first time, empirical results show the effectiveness of the greedy approach of Marchand and Shawe-Taylor Marchand and Shawe-

Taylor [2002] at building SCM of both small compression set and empirical risk. This is a very surprising result given the simplicity and the low complexity of the greedy algorithm.

Bibliographie

T. Achterberg. *SCIP-a framework to integrate constraint and mixed integer programming.* Konrad-Zuse-Zentrum für Informationstechnik, 2004.

Reka Albert. Scale-free networks in cell biology. *Journal of cell science*, 118(21) :4947–4957, 2005.

Prasanna Alluri, Bo Liu, Peng Yu, Xiangshu Xiao, and Thomas Kodadek. Isolation and characterization of coactivator-binding peptoids from a combinatorial library. *Molecular BioSystems*, 2(11) :568–579, 2006.

Prasanna G Alluri, M Muralidhar Reddy, Kiran Bachhawat-Sikder, Hernando J Olivos, and Thomas Kodadek. Isolation of protein ligands from large peptoid libraries. *Journal of the American Chemical Society*, 125(46) :13995–14004, 2003.

Luca Baldassarre, Lorenzo Rosasco, Annalisa Barla, and Alessandro Verri. Multi-output learning via spectral filtering. *Machine Learning*, 87 :259–301, 2012.

Pierre Baldi and Søren Brunak. *Bioinformatics : the machine learning approach.* MIT press, 2001.

Asa Ben-Hur and William Stafford Noble. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(suppl 1) :i38–i46, 2005.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

Andrew J. Bordner and Hans D. Mittelmann. Multirta : A simple yet reliable method for predicting peptide binding affinities for multiple class ii mhc allotypes. *BMC Bioinformatics*, 11 :482, 2010a.

Andrew J Bordner and Hans D Mittelmann. Prediction of the binding affinities of peptides to class II MHC using a regularized thermodynamic model. *BMC Bioinformatics*, 11(1) :41, 2010b. ISSN 1471-2105. doi : 10.1186/1471-2105-11-41. URL www.biomedcentral.com/1471-2105/11/41.

A.P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7) :1145–1159, 1997.

Andrew William Brander and Mark C Sinclair. *A comparative study of k-shortest path algorithms*. PhD thesis, Citeseer, 1995.

Céline Brouard, Florence D'Alche-Buc, and Marie Szafranski. Semi-supervised penalized output kernel regression for link prediction. In *International Conference on Machine Learning (ICML)*, 2011.

Andrea Caponnetto and Ernesto De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3) :331–368, 2007.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3) :273–297, 1995.

Corinna Cortes, Mehryar Mohri, and Jason Weston. A general regression framework for learning string-to-string mappings. In Gökhan Bakır, Thomas Hofmann, Bernhard Schölkopf, Alexander J. Smola, Ben Taskar, and S. Vishwanathan, editors, *Predicting Structured Data*, chapter 8, pages 143–168. MIT Press, Cambridge, MA, 2007.

Luca Costantino and Daniela Barlocco. Privileged structures as leads in medicinal chemistry. *Current medicinal chemistry*, 13(1) :65–85, 2006.

Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

Jiri Damborsky and Jan Brezovsky. Computational tools for designing and engineering biocatalysts. *Current opinion in chemical biology*, 13(1) :26–34, 2009.

Dana-Farber Cancer Institute. 2nd machine learning competition in immunology, November 2012. URL `http://bio.dfci.harvard.edu/DFRMLI/HTML/natural.php`.

Karin E de Visser, Alexandra Eichten, and Lisa M Coussens. Paradoxical roles of the immune system during cancer development. *Nature reviews cancer*, 6(1) :24–37, 2006.

Alexander Dömling. Small molecular weight protein–protein interaction antagonists—an insurmountable challenge ? *Current opinion in chemical biology*, 12(3) :281–291, 2008.

Jannick Dyrløv Bendtsen, Henrik Nielsen, Gunnar von Heijne, and Søren Brunak. Improved prediction of signal peptides : Signalp 3.0. *Journal of molecular biology*, 340(4) :783–795, 2004.

Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220. ACM, 2008.

David Eppstein. Finding the k shortest paths. *SIAM Journal on computing*, 28(2) :652–673, 1998.

Jean-Loup Faulon, Milind Misra, Shawn Martin, Ken Sale, and Rajat Sapra. Genome scale enzyme–metabolite and drug–target interaction predictions using the signature molecular descriptor. *Bioinformatics*, 24(2) :225–233, 2008.

David Filmore. It's a gpcr world. *Modern drug discovery*, 7 :24–28, 2004.

Sally Floyd and Manfred Warmuth. Sample compression, learnability, and the vapnik-chervonenkis dimension. *Machine Learning*, 21 :269–304, 1995. ISSN 0885-6125.

David C Fry. Protein–protein interactions as targets for small molecule drug discovery. *Peptide Science*, 84(6) :535–552, 2006.

Thomas Gärtner and Shankar Vembu. On structured output training : hard cases and an efficient alternative. *Machine Learning*, 76(2-3) :227–242, 2009.

Pascal Germain, Alexandre Lacasse, François Laviolette, and Mario Marchand. PAC-Bayesian learning of linear classifiers. In *International Conference on Machine Learning (ICML)*, 2009.

Pascal Germain, Alexandre Lacoste, François Laviolette, Mario Marchand, and Sara Shanian. A PAC-Bayes sample-compression approach to kernel methods. In *International Conference on Machine Learning (ICML)*, 2011.

Sébastien Giguère, Alexandre Drouin, Alexandre Lacoste, Mario Marchand, Jacques Corbeil, and François Laviolette. Mhc-np : Predicting peptides naturally processed by the mhc. *Journal of Immunological Methods*, 2013a.

Sébastien Giguère, François Laviolette, Mario Marchand, and Khadidja Sylla. Risk bounds and learning algorithms for the regression approach to structured output prediction. In *International Conference on Machine Learning (ICML)*, 2013b.

Sébastien Giguère, Mario Marchand, François Laviolette, Alexandre Drouin, and Jacques Corbeil. Learning a peptide-protein binding affinity predictor with kernel ridge regression. *BMC Bioinformatics*, 14, 2013.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning : data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2) :83–85, 2005.

Brice Hoffmann, Mikhail Zaslavskiy, Jean-Philippe Vert, and Véronique Stoven. A new protein binding pocket similarity measure based on comparison of clouds of atoms in 3d : application to ligand prediction. *BMC bioinformatics*, 11(1) :99, 2010.

Victor J Hruby and Minying Cai. Design of peptide and peptidomimetic ligands with novel pharmacological activity profiles. *Annual review of pharmacology and toxicology*, 53 :557–580, 2013.

Martial Hue, Michael Riffle, Jean-Philippe Vert, and William S Noble. Large-scale prediction of protein-protein interactions from structures. *BMC bioinformatics*, 11(1) :144, 2010.

Ruth Huey, Garrett M Morris, Arthur J Olson, and David S Goodsell. A semiempirical free energy force field with charge-based desolvation. *Journal of computational chemistry*, 28 (6) :1145–1152, 2007.

Zakria Hussain, Sandor Szedmak, and John Shawe-Taylor. The linear programming set covering machine. 2004.

Laurent Jacob and Jean-Philippe Vert. Efficient peptide–mhc-i binding prediction for alleles with few known binders. *Bioinformatics*, 24(3) :358–366, 2008.

Laurent Jacob, Brice Hoffmann, Véronique Stoven, and Jean-Philippe Vert. Virtual screening of gpcrs : an in silico chemogenomics approach. *BMC bioinformatics*, 9(1) :363, 2008.

Eric A Jamois. Reagent-based and product-based computational approaches in library design. *Current opinion in chemical biology*, 7(3) :326–330, 2003.

M Jesus Perez de Vega, Mercedes Martin-Martinez, and Rosario Gonzalez-Muniz. Modulation of protein-protein interactions by stabilizing/mimicking protein secondary structure elements. *Current topics in medicinal chemistry*, 7(1) :33–62, 2007.

Jing Jiang. A literature survey on domain adaptation of statistical classifiers. *URL : http ://sifaka.cs.uiuc.edu/jiang4/domainadaptation/survey*, 2008.

T. Joachims. Transductive inference for text classification using support vector machines. *International Conference on Machine Learning (ICML)*, pages 200–209, 1999.

Sang Hoon Joo and Dehua Pei. Synthesis and screening of support-bound combinatorial peptide libraries with free c-termini : Determination of the sequence specificity of pdz domains†. *Biochemistry*, 47(9) :3061–3072, 2008.

William L Jorgensen. Rusting of the lock and key model for protein-ligand binding. *Science*, 254(5034) :954–955, 1991.

Hachem Kadri, Mohammad Ghavamzadeh, and Philippe Preux. A generalized kernel approach to structured output learning. In *International Conference on Machine Learning (ICML)*, 2013.

Rajeev Kharb, Meenakshi Rana, Prabodh Chander Sharma, and Mohammad Shahar Yar. Therapeutic importance of peptidomimetics in medicinal chemistry. *Journal of Chemical and Pharmaceutical Research*, 3(6) :173–186, 2011.

Pappanaicken R Kumaresan, Yan Wang, Mary Saunders, Yoshiko Maeda, Ruiwu Liu, Xiaobing Wang, and Kit Sang Lam. Rapid discovery of death ligands with one-bead-two-compound combinatorial library methods. *ACS combinatorial science*, 13(3) :259–264, 2011.

Alexandre Lacoste, Mario Marchand, François Laviolette, and Hugo Larochelle. Agnostic bayesian learning of ensembles. In *Proceedings of The 31st International Conference on Machine Learning*, pages 611–619, 2014.

Kit S Lam, Michal Lebl, and Viktor Krchnák. The "one-bead-one-compound" combinatorial library method. *Chemical reviews*, 97(2) :411–448, 1997.

John Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6 :273–306, 2005.

Gniewomir Latacz, E Pekala, A Ciopinska, and K Kiec-Kononowicz. Unnatural d-amino acids as building blocks of new peptidomimetics. *Acta Poloniae Pharmaceutica–Drug Research*, 62 :430–433, 2006.

Martin Latterich and Jacques Corbeil. Label-free detection of biomolecular interactions in real time with a nano-porous silicon-based detection method. *Proteome science*, 6(1) :31, 2008.

François Laviolette and Mario Marchand. PAC-Bayes risk bounds for stochastic averages and majority votes of sample-compressed classifiers. *Journal of Machine Learning Research*, 8 : 1461–1487, 2007.

Eugene L Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18(7) : 401–405, 1972.

Christina S Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel : A string kernel for svm protein classification. In *Pacific symposium on biocomputing*, volume 7, pages 566–575. World Scientific, 2002.

Christina S Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4) : 467–476, 2004.

Tao Liu, Sang Hoon Joo, Jeffrey L Voorhees, Charles L Brooks, and Dehua Pei. Synthesis and screening of a cyclic peptide library : discovery of small-molecule ligands against human prolactin receptor. *Bioorganic & medicinal chemistry*, 17(3) :1026–1033, 2009.

Tao Liu, Ziqing Qian, Qing Xiao, and Dehua Pei. High-throughput screening of one-bead-one-compound libraries : identification of cyclic peptidyl inhibitors against calcineurin/nfat interaction. *ACS combinatorial science*, 13(5) :537–546, 2011.

C. Lundegaard, K. Lamberth, M. Harndahl, S. Buus, O. Lund, and M. Nielsen. Netmhc-3.0 : accurate web accessible predictions of human, mouse and monkey mhc class i affinities for peptides of length 8-11. *Nucleic acids research*, 36(Web Server issue) :W509–512, 2008.

R.M.V.M.I. Lynce. Parallel search for boolean optimization. 2011.

V. Manquinho and J.P. Marques-Silva. On using cutting planes in pseudo-boolean optimization. *Journal on Satisfiability, Boolean Modeling and Computation*, 2 :209–219, 2006.

Mario Marchand and John Shawe-Taylor. The set covering machine. *Journal of Machine Learning Research*, 3 :723–746, 2002.

Mario Marchand and Marina Sokolova. Learning with decision lists of data-dependent features. *J. Mach. Learn. Res.*, 6 :427–451, December 2005. ISSN 1532-4435.

María C Martínez-Ceron, Mariela M Marani, Marta Taulés, Marina Etcheverrigaray, Fernando Albericio, Osvaldo Cascone, and Silvia A Camperi. Affinity chromatography based on a combinatorial strategy for rerythropoietin purification. *ACS combinatorial science*, 13(3) : 251–258, 2011.

Andreas Maurer. A note on the PAC Bayesian theorem. *CoRR*, cs.LG/0411099, 2004.

David McAllester. Generalization bounds and consistency for structured labeling. In Gökhan Bakır, Thomas Hofmann, Bernhard Schölkopf, Alexander J. Smola, Ben Taskar, and S. Vishwanathan, editors, *Predicting Structured Data*, chapter 11, pages 247–261. MIT Press, Cambridge, MA, 2007.

David A McAllester. PAC-Bayesian stochastic model selection. *Machine Learning*, 51(1) : 5–21, 2003.

ROGER P MEE, TIM R AUTON, and PHILLIP J MORGAN. Design of active analogues of a 15-residue peptide using d-optimal design, qsar and a combinatorial search algorithm. *The Journal of peptide research*, 49(1) :89–102, 1997.

P. Meinicke, M. Tech, B. Morgenstern, and R. Merkl. Oligo kernels for datamining on biological sequences : A case study on prokaryotic translation initiation sites. *BMC Bioinformatics*, 5, 2004.

Asher Mullard. Protein–protein interaction inhibitors get into the groove. *Nature Reviews Drug Discovery*, 11(3) :173–175, 2012.

Nobuyoshi Nagamine and Yasubumi Sakakibara. Statistical prediction of protein–chemical interactions based on chemical structure and mass spectrometry data. *Bioinformatics*, 23 (15) :2004–2012, 2007.

Morten Nielsen, Claus Lundegaard, Thomas Blicher, Bjoern Peters, Alessandro Sette, Sune Justesen, Søren Buus, and Ole Lund. Quantitative predictions of peptide binding to any hla-dr molecule of known sequence : Netmhciipan. *PLoS computational biology*, 4(7) :e1000107, 2008.

Morten Nielsen, Sune Justesen, Ole Lund, Claus Lundegaard, and Søren Buus. Netmhciipan-2.0-improved pan-specific hla-dr predictions using a novel concurrent alignment and weight optimization training procedure. *Immunome research*, 6(1) :9, 2010.

Angel R. Ortiz, Charlie E.M. Strauss, and Osvaldo Olmea. Mammoth (matching molecular models obtained from theory) : An automated method for model comparison. *Protein Science*, 11(11) :2606–2621, 2002. ISSN 1469-896X. doi : 10.1110/ps.0215902.

John P Overington, Bissan Al-Lazikani, and Andrew L Hopkins. How many drug targets are there ? *Nature reviews Drug discovery*, 5(12) :993–996, 2006.

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 É. Duchesnay. Scikit-learn : Machine learning in python. *Journal of Machine Learning Research*, 12 :2825–2830, 2011.

B. Peters, J. Sidney, P. Bourne, H.H. Bui, S. Buus, G. Doh, W. Fleri, M. Kronenberg, R. Kubo, O. Lund, D. Nemazee, J.V. Ponomarenko, M. Sathiamurthy, S. Schoenberger, S. Stewart, P. Surko, S. Way, S. Wilson, and A. Sette. The immune epitope database and analysis resource : from vision to blueprint. *PLoS biology.*, 3(3) :e91, 2005.

Stephen D Pickett, Iain M McLay, and David E Clark. Enhancing the hit-to-lead properties of lead optimization libraries. *Journal of chemical information and computer sciences*, 40 (2) :263–272, 2000.

J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3) :61–74, 1999.

J. Qiu, M. Hue, A. Ben-Hur, J.-P. Vert, and W.S. Noble. A structural alignment kernel for protein structures. *Bioinformatics*, 23(9) :1090–1098, 2007.

C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation And Machine Learning. Mit Press, 2006. ISBN 9780262182539.

Gunnar Rätsch and Sören Sonnenburg. Accurate Splice Site Detection for Caenorhabditis elegans. In B and J. P. Vert, editors, *Kernel Methods in Computational Biology*, pages 277–298. MIT Press, 2004.

J. Robinson, A. Malik, P. Parham, J.G. Bodmer, and S.G.E. Marsh. Imgt/hla database – a sequence database for the human major histocompatibility complex. *Tissue Antigens*, 55(3) :280–287, 2000. ISSN 1399-0039. doi : 10.1034/j.1399-0039.2000.550314.x. URL http://dx.doi.org/10.1034/j.1399-0039.2000.550314.x.

Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *The Journal of Machine Learning Research*, 7 :1601–1626, 2006.

H. Saigo, J.-P. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11) :1682–1689, 2004.

Gisbert Schneider. Virtual screening : an endless staircase ? *Nature Reviews Drug Discovery*, 9(4) :273–276, 2010.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

Laurel O Sillerud and Richard S Larson. Design and structure of peptide and peptidomimetic antagonists of protein-protein interaction. *Current Protein and Peptide Science*, 6(2) :151–169, 2005.

Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3) :199–222, 2004.

S Joshua Swamidass, Jonathan Chen, Jocelyne Bruand, Peter Phung, Liva Ralaivola, and Pierre Baldi. Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics*, 21(suppl 1) :i359–i368, 2005.

John A Swets. Measuring the accuracy of diagnostic systems. *Science*, 240(4857) :1285–1293, 1988.

Masataka Takarabe, Masaaki Kotera, Yosuke Nishimura, Susumu Goto, and Yoshihiro Yamanishi. Drug target prediction using adverse event report systems : a pharmacogenomic approach. *Bioinformatics*, 28(18) :i611–i618, September 2012. doi : 10.1093/bioinformatics/bts413. URL http://dx.doi.org/10.1093/bioinformatics/bts413.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

Peter L Toogood. Inhibition of protein-protein association by small molecules : approaches and progress. *Journal of medicinal chemistry*, 45(8) :1543–1558, 2002.

Nora Toussaint, Christian Widmer, Oliver Kohlbacher, and Gunnar Rätsch. Exploiting physico-chemical properties in string kernels. *BMC bioinformatics*, 11(Suppl 8) :S7, 2010.

Nora C Toussaint and Oliver Kohlbacher. Towards in silico design of epitope-based vaccines. *Expert Opinion on Drug Discovery*, 2009.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6 : 1453–1484, 2005.

Jan G.R. Ufkes, Berend J. Visser, Gerritdina Heuver, Herman J. Wynne, and Cornelis Van Der Meer. Further studies on the structure-activity relationships of bradykinin-potentiating peptides. *European Journal of Pharmacology*, 79(1–2) :155 – 158, 1982.

Peter Vanhee, Joke Reumers, Francois Stricher, Lies Baeten, Luis Serrano, Joost Schymkowitz, and Frederic Rousseau. Pepx : a structural database of non-redundant protein–peptide complexes. *Nucleic acids research*, 38(suppl 1) :D545–D551, 2010.

Peter Vanhee, Almer M van der Sloot, Erik Verschueren, Luis Serrano, Frederic Rousseau, and Joost Schymkowitz. Computational design of peptide ligands. *Trends in biotechnology*, 29 (5) :231–239, 2011.

David Wade and Jukka Englund. Synthetic antibiotic peptides database. *Protein and peptide letters*, 9(1) :53–57, 2002.

Binqing Q Wei, Larry H Weaver, Anna M Ferrari, Brian W Matthews, and Brian K Shoichet. Testing a flexible-receptor docking algorithm in a model binding site. *Journal of molecular biology*, 337(5) :1161–1182, 2004.

Donald A Wellings and Eric Atherton. [4] standard fmoc protocols. *Methods in enzymology*, 289 :44–67, 1997.

James A Wells and Christopher L McClendon. Reaching for high-hanging fruit in drug discovery at protein–protein interfaces. *Nature*, 450(7172) :1001–1009, 2007.

Irith Wiegand, Kai Hilpert, and Robert EW Hancock. Agar and broth dilution methods to determine the minimal inhibitory concentration (mic) of antimicrobial substances. *Nature protocols*, 3(2) :163–175, 2008.

Jin Y Yen. Finding the k shortest loopless paths in a network. *management Science*, 17(11) : 712–716, 1971.

L. Zhang, K. Udaka, H. Mamitsuka, and S. Zhu. Toward more accurate pan-specific mhc-peptide binding prediction : A review of current methods and tools. *Briefings in Bioinformatics*, 13(3) :350–364, 2012.

Tong Zhang. Information theoretical upper and lower bounds for statistical estimation. *IEEE Transaction on Information Theory*, 52 :1307–1321, 2006.

Yanyan Zhang, Shanggen Zhou, Anne-Sophie Wavreille, James DeWille, and Dehua Pei. Cyclic peptidyl inhibitors of grb2 and tensin sh2 domains identified from combinatorial libraries. *Journal of combinatorial chemistry*, 10(2) :247–255, 2008.

Peng Zhou, Xiang Chen, Yuqian Wu, and Zhicai Shang. Gaussian process : an alternative approach for qsam modeling of peptides. *Amino Acids*, 38(1) :199–212, 2010.