UNIVERSITÉ DU QUÉBEC À MONTRÉAL

NOUVELLES APPROCHES DE CLASSIFICATION BINAIRE BASÉES SUR LE SÉPARATEUR À VASTE MARGE POUR LES DONNÉES DE GRANDE DIMENSION

THÈSE

PRÉSENTÉE

COMME EXIGENCE PARTIELLE

DU DOCTORAT EN MATHÉMATIQUES

PAR

RACHID KHAROUBI

JUILLET 2024

REMERCIEMENTS

La rédaction de cette thèse n'aurait pas été possible sans le précieux soutien d'un nombre de personnes. D'abord, je tiens à remercier mon directeur de recherche, Karim Oualkacha, qui m'a soutenu et qui m'a encouragé à suivre mes études en doctorat après l'obtention de mon diplôme de maîtrise avec lui. J'ai beaucoup appris en échangeant avec lui lors des discussions et lors des rencontres. Ensuite, je remercie également mon codirecteur, Abdallah Mkhadri, pour ses conseils et son soutien. J'ai été ravi de travailler avec une équipe de recherche de grande qualité. Mes vifs remerciements à vous deux. Finalement, je veux remercier mes parents au Maroc et ma petite famille ici au Canada pour leur soutien. Je tiens aussi à remercier mon épouse, Bella Fatima, qui m'a cru et qui m'a encouragée tout au long du parcours de mes études.

DÉDICACE

À mes parents qui m'ont éduqué,

à ma femme qui m'a encouragé et m'a soutenu dans mon projet,

et à tous mes enfants.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

Le Séparateur à Vaste Marge (SVM) est un outil d'apprentissage statistique supervisé issu d'une généralisation des classificateurs linéaires. Il a suscité beaucoup d'intérêt dans l'analyse discriminante et le classement des données en raison de son pouvoir prédictif et sa capacité à résoudre les problèmes de classement binaire en présence des données de grande dimension. De nombreuses approches liées à ce classificateur ont été élaborées dans ce domaine de recherche très animé. Cette thèse propose des généralisations de l'apporche SVM pour le classement binaire en présence des données volumineuses. Les méthodes proposées visent l'amélioration de la prédiction, le classement des données et la sélection de variables.

Nous proposons d'abord la méthode CCNSVM [de l'anglais Cluster Correlation-Network SVM]. Cette approche a pour but d'identifier/selectionner des groupes de prédicteurs importants pour une variable réponse binaire et d'estimer leurs coefficients de régression, de façon simultanée, dans un modèle SVM pénalisé. La méthode CCNSVM exploite la structure de corrélation entre les prédicteurs afin de les regrouper dans des groupes homogènes en se basant sur une mesure capturée dans une matrice de similarité, nomée TOM [de l'anglais Topological Overlap Matrix]. La pénalité CCN exploite de tels regroupements des prédicteurs afin de selectionner les goupes des prédicteurs qui sont associés avec la variable réponse (c.-à-d. les groupes qui sont importants pour distinguer davantage les deux classes de la variable réponse binaire).

Ensuite, nous proposons la fonction de perte BernSVM, basée sur les polynômes de Bernstein, qui est une approximation lisse de la fonction de perte SVM . En effet, la fonction de perte SVM n'est pas suffisamment dérivable, ce qui rend le problème d'optimisation sous-jacent difficile à résoudre. Ainsi, la fonction de perte BernSVM permet l'estimation des paramètres du modèle SVM pénalisé de façon éfficace en utilisant des techniques d'optimisation standard comme l'algorithme de la descente par coordonnée et/ou celui des moindres carrés repondérés itératifs (IRLS en anglais). Les propriétés de la fonction de perte BernSVM permettent aussi de développer le comportement (théorique) non asymptotique des estimateurs des paramètres de la méthode SVM régularisée.

Nous proposons finalement la méthode SplitSVM qui modélise la relation entre la variable réponse et les prédicteurs via un ensemble de modèles SVM. En effet, SplitSVM consiste à employer plusieurs modèles BernSVM pénalisés au lieu d'en considérer un seul afin d'exploiter la diversité entre ces modèles, en prenant en compte différentes combinaisons des variables explicatives dans chaque modèle. Une fois tous les modèles estimés, ils sont agrégés dans un modèle final afin de mieux charactériser la relation/association entre la variable réponse et les prédicteurs.

Les résultats des études de simulation et l'analyse des données réelles montrent la performance et l'efficacité des approches proposées dans les chapitres 2, 3 et 4, comparées aux compétiteurs existants dans ce domaine d'apprentissage supervisé.

INTRODUCTION

L'apprentissage statistique est une branche de la statistique qui se concentre sur le développement et l'utilisation des outils statistiques qui nous aident à analyser et à mieux comprendre les données. Ce type d'apprentissage peut être supervisé ou non supervisé. L'apprentissage supervisé permet de développer un modèle statistique prédictif en modélisant une variable réponse $\boldsymbol{Y}$ en fonction de p variables explicatives regroupées dans une matrice $\boldsymbol{X}$ d'ordre $n \times p$, où n est la taille de l'échantillon. Ce genre d'apprentissage peut être utilisé dans différents domaines, entre autres, la finance, la médecine et l'économie. L'apprentissage non supervisé consiste à trouver les relations, les structures et le regroupement de plusieurs variables de $\boldsymbol{X}$ collectées sur un ensemble de n sujets. Les travaux de recherche de ma thèse se situent dans le cadre de l'apprentissage supervisé en présence de données de grande dimension où p est plus grand que n.

Supposons que nous possédons un jeu de données de n observations pour une variable réponse $\boldsymbol{Y}$ et p variables explicatives (prédicteurs) $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_p$ sachant que le nombre de ces derniers est plus grand que le nombre d'observations. L'objectif est de construire des modèles statistiques dont le but est d'améliorer la prédiction et le classement. La variable réponse peut être quantitative ou catégorielle. Si la variable réponse est catégorielle avec deux classes, disons 1 et -1, alors la prédiction d'une observation fait référence à classer cette observation à l'une des deux classes 1 ou -1. Il y a plusieurs techniques de classement qui peuvent être utilisées afin de prédire la classe des observations. De telles techniques se basent sur des règles de classement afin de minimiser le coût (espéré) de la mauvaise classification. On peut citer, entre autres, la régression logistique Hosmer Jr *et al.* (2013), l'analyse discriminante linéaire ou quadratique Hastie *et al.* (2009) et les séparateurs à vaste marge [SVM, de l'anglais Support Vector Machines]. La méthode SVM est une technique d'apprentissage statistique très populaire dans le domaine de la classification. Ce classificateur a été présenté pour la première fois par Cortes et Vapnik (1995) et Vapnik (1999) et depuis la méthode est devenue populaire pour ses bonnes performances pour analyser des données de grande dimension comparée aux autres méthodes existantes de classement.

Vue comme méthode de régularisation, l'approche SVM classique est difficile à interpréter puisque tous les prédicteurs sont utilisés pour construire la règle de classement (c.-à-d. construire le plan séparateur). En effet, le problème d'optimisation sous-jaçant de la méthode SVM peut se reformuler

en un problème d'otpimisation impliquant la fonction de perte SVM et une pénalité de la norme quadratique L_2 du vecteur des paramètres du modèle. En présence de données de grande dimension, plusieurs prédicteurs (bruits) pourraient être non pertinents pour la classification. Par construction, la méthode SVM standard n'effectue pas la sélection des prédicteurs pertinents (c.-à-d. elle estime tous les paramètres du modèle). Ceci implique un rétrécissement systématique, vers l'origine, de tous les estimateurs des coefficients du modèle. Donc, la méthode SVM standard ne produit pas des modèles parcimonieux, faciles à interpréter et qui peuvent se généraliser à la population globale de la variable réponse. Par conséquent, la notion de pénalité est introduite dans le cadre de l'approche SVM pour remédier à ce problème. Ainsi, d'autres alternatives à la méthode SVM standard, basées sur des pénalités, ont été développées et largement utilisées dans la pratique pour permettre à la fois l'estimation et la sélection de variables.

Parmi les approches SVM pénalisées permettant simultanément la sélection et l'estimation, nous trouvons la méthode SVM avec la pénalité Lasso (ℓ_1SVM ; Bradley et Mangasarian (1998) ;Zhu *et al.* (2003)), SVM avec la pénalité Elastique Net (EN) (ENSVM ; Wang *et al.* (2006), SVM avec pénalité non convexe SCAD (SCADSVM ; Zhang *et al.* (2016)) et la méthode SVM avec la combinaison de la pénalité SCAD et ℓ_2 (SnetSVM), Becker *et al.* (2011). La pénalité EN produit un modèle parcimonieux en rétrécissant tous les coefficients vers l'origine. Toutefois, dans certains contextes, ce comportement peut ne pas être souhaitable : si certaines variables sont fortement corrélées entre elles et associées à la variable réponse. Cependant, Witten *et al.* (2014) ont proposé une nouvelle pénalité appelée CEN [de l'anglais Cluster Elastic Net] dans le cadre de la régression linéaire en grande dimension afin de regrouper les prédicteurs importants dans des groupes sachant que les estimations des coefficients des prédicteurs qui forment chaque groupe sont similaires. Ainsi, le comportement de la pénalité CEN est différent de celle de la pénalité EN. Les groupes estimés sont utilisés pour améliorer la prédiction. Dans le chapitre 2, nous allons proposer CCN, une nouvelle pénalité, similaire à CEN, pour améliorer la prédiction et la sélection de variables pour la méthode SVM en grande dimension.

La fonction de perte SVM n'est pas différentiable en 1, de sorte que les techniques d'optimisation standard ne peuvent pas être appliquées directement. Pour surmonter ce problème, des alternatives récentes sont développées sur la base de l'idée principale d'approximer la fonction de perte SVM avec une fonction lisse qui est différentiable partout. Nous citons, par exemple, la fonction de perte Huber Zou et Li (2008), considérée par Yang et Zou (2013) et Kharoubi *et al.* (2019). Dans ces deux travaux, les auteurs ont remplacé la fonction de perte SVM par la fonction de perte Huber pour pouvoir

approximer la fonction objective globale de la méthode SVM par une fonction quadratique. Cette dernière permet l'implémentation d'un algorithme efficace et rapide, qui combine les deux techniques, Maximisation-Minimisation (MM) Hunter et Lange (2004) et la Descente par Coordonée (DC) Tseng (2001). Dans le chapitre 3, nous allons proposer une nouvelle approximation de la fonction de perte SVM, basée sur les polynômes de Bernstein Lorentz (2013). Cette fonction de perte est deux fois continue et dérivable, ce qui facilite la résolution du problème d'optimisation sous-jaçent de la méthode SVM en utilisant des techniques algorithmiques standard comme la descente par gradient, Newton-Raphson, CD, MM et d'autres.

Dans la modélisation statistique moderne, l'apprentissage ensembliste est une approche qui a pour but d'analyser les données en utilisant plusieurs modèles simultanément et les agréger ensuite pour améliorer le pouvoir prédictif. Dans le cadre de la régression linéaire, Christidis *et al.* (2020) ont proposé une méthode ensembliste basée sur une pénalité qui balance la parcimonie des modèles individuels et la diversité entre les modèles différents. Christidis *et al.* (2021) ont généralisé le même concept pour les modèles linéaires généralisés (GLM). Dans le chapitre 4 de la thèse, nous allons adapter la même méthodologie pour proposer une nouvelle méthode ensembliste pour la méthode SVM afin d'améliorer le pouvoir prédictif de ce classificateur et la sélection de variables.

En résumé, cette thèse s'adresse à améliorer la sélection de variables et le pouvoir prédictif du classificateur SVM à l'aide de nouvelles approximations et nouvelles pénalités.

Cette thèse est composée de trois contributions développés dans les chapitres 2, 3 et 4. Dans le chapitre 2 (première contribution), nous proposons le CCNSVM, une méthode de classification binaire qui combine la méthode SVM avec la pénalité CCN. Cette dernière est une nouvelle pénalité pour régulariser la méthode SVM. Nous avons remplacé la fonction de perte SVM avec la fonction de perte Huber pour pouvoir majorer la fonction objective globale par une fonction quadratique facile à minimiser. L'algorithme proposé pour estimer les paramètres du modèle SVM pénalisé est un algorithme de type minimisation alternée Hardt (2014). D'abord, nous estimons la structure de groupe dans les prédicteurs avec l'algorithme K-means Likas *et al.* (2003). Ensuite, nous combinons la descente par coordonnée et le principe de majoration-minimisation (MM) pour estimer les chemins de solutions pour les estimateurs des paramètres du modèle.

Dans le chapitre 3 (deuxième contribution), nous proposons la fonction de perte BernSVM, une fonction de perte lisse et suffisamment dérivable qui approxime la fonction de perte SVM. Elle est

développée avec une technique d'approximation d'une fonction à l'aide des polynômes de Bernstein. Nous proposons deux algorithmes pour résoudre le modèle BernSVM pénalisé. Le premier algorithme combine la descente par coordonnée et le principe MM. Cet algorithme est similaire à l'algorithme proposé dans le chapitre 2. Le deuxième algorithme combine la méthode IRLS et la descente par coordonnée. Nous utilisons les propriétés intéressantes de la fonction de perte BernSVM (c-à-d. une fonction deux fois dérivable partout) pour dériver le comportement (théorique) non asymptotique des estimateurs des coefficients du modèle proposé. En effet, nous dérivons une borne supérieure du carré de la différence entre les estimateurs des coefficients et les vrais coefficients.

Dans le chapitre 4 (troisième contribution), nous proposons SplitSVM, une nouvelle approche de l'apprentissage ensembliste. En effet, un ensemble de modèles différents de la méthode SVM sont considérés, afin de prédire une variable réponse binaire, en présence de données de grande dimension. Les paramètres des modèles proposés par cette méthode sont estimés simultanément, et ils sont agrégés, ensuite, dans un estimateur global. Ce dernier est obtenu comme une moyenne des estimateurs des coefficients de tous les modèles considérés. Cette méthode exploite deux pénalités, le Lasso pour avoir des modèles parcimonieux et une pénalité de diversité pour favoriser la diversité entre les modèles qui lient les covariables à la variable réponse binaire. Nous développons également une borne supérieure du carré de la différence entre l'estimateur global ensembliste et les coefficients du vrai modèle qui lie la variable réponse avec les prédicteurs.

Toute la méthodologie développée dans la thèse est appuyée par des études exhaustives de simulation et des analyses de données réelles de grande dimension.

CHAPITRE 1
PRÉLIMINAIRE

Ce chapitre élabore les concepts de base sur le classement binaire afin que le lecteur puisse comprendre les chapitres suivants de la thèse. Nous allons d'abord rappeler plusieurs fonctions de pertes utilisées dans les problèmes de classement des données, présentes dans la littérature. Nous allons introduire la méthode SVM dans le cadre de l'apprentissage statistique en présence de données volumineuses. Nous décrivons également les différentes méthodes de régularisation utilisées dans la littérature afin d'obtenir des modèles faciles à interpréter, des modèles avec des propriétés souhaitées, comme la parcimonie, le compromis biais/variance et la sélection des variables importantes.

1.1 Aperçu général sur les fonctions de pertes utilisées pour le classement des données

Dans un contexte de classement binaire, nous possédons un ensemble de données d'entrainement $(\boldsymbol{x}_i, y_i)$, avec $\boldsymbol{x}_i \in \mathbb{R}^p$ et $y_i \in \{-1, 1\}$, avec $i = 1, ..., n$. L'objectif des méthodes de classement est de construire une règle de classement qui est basée sur une fonction de perte afin de minimiser l'erreur de classement, c.-à-d. assigner une observation $\boldsymbol{x}_i$ à la bonne classe. Pour atteindre cet objectif, nous allons définir d'abord l'hyperplan séparateur, H, qui sépare les données en deux classes 1 et -1 (c-à-d l'hyperplan séparateur, H, qui sépare les sujets avec une valeur de $y = 1$ et les sujets avec $y = -1$). L'hyperplan H est présenté par l'équation suivante : $H = \{\boldsymbol{x} \in \mathbb{R}^p | h(\boldsymbol{x}) = 0\}$, où la fonction $h(.)$ est définie par

$$h(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\beta} + \beta_0.$$

Ainsi, une règle de décision induite par $h(\boldsymbol{x})$ est donnée par $sign(h(\boldsymbol{x}))$. Si les données sont linéairement séparables, l'hyperplan H, va séparer l'espace en deux régions, et chaque région détermine l'une des deux classes 1 ou -1. Dans la section suivante, nous allons introduire quelques fonctions de pertes utilisées en classement binaire. D'abord, nous définissons le concept de la marge. Soit $\boldsymbol{m} = (m_1, m_2, \ldots, m_n)^\top$, un vecteur de $\mathbb{R}^n$. La marge pour l'observation i est donnée par

$$m_i = y_i h(\boldsymbol{x}_i). \tag{1.1}$$

Il est à noter que même si en général une règle de classement (c-à-d. assigner une observation $\boldsymbol{x}_i$ au groupe 1 ou -1) se fait sur la base du signe de la marge m_i, comme nous allons le constater plus tard dans ce chapitre, le plus important dans ce type de classification binaire c'est plutôt le signe

de la fonction $h(\boldsymbol{x}_i)$. En effet, on peut montrer que la distance (signée) de tout $\boldsymbol{x}_i$ par rapport à l'hyperplan H peut s'écrire comme $d(\boldsymbol{x}_i, H) := h(\boldsymbol{x}_i)/||h^{'}(\boldsymbol{x}_i)||$, où $h^{'}(\boldsymbol{x}) = \partial h(\boldsymbol{x})/\partial \boldsymbol{x}$. Ainsi, on constate que $h(\boldsymbol{x}_i)$ est proportionnelle à la distance signée de $\boldsymbol{x}_i$ par rapport à l'hyperplan défini par $h(\boldsymbol{x}) = 0$. Puisque H est un hyperplan qui sépare les deux classes, on peut déduire que le signe de $h(\boldsymbol{x}_i)$ est le même pour toutes les observations d'une même classe. Nous avons également que $h(\boldsymbol{x}_i)$ et $h(\boldsymbol{x}_{i'})$ ont deux signes opposés lorsque $\boldsymbol{x}_i$ et $\boldsymbol{x}_{i'}$ n'appartiennent pas à une même classe.

1.1.1 La fonction de perte zéro-un

Dans le cadre de la régression linéaire, où la variable réponse y est continue, nous nous intéressons aux résidus, $\epsilon_i = y_i - h(\boldsymbol{x}_i)$, afin de construire la fonction de perte quadratique. Par similarité, dans le cadre de la classification, nous nous intéressons aux marges m_i. Cependant, nous avons besoin, tout d'abord, de définir une règle de classification. Pour ce faire, rappelons que c'est le signe de $h(x)$ qui distingue les observations dans les deux classes. Ainsi, la règle de classement pourrait être : assigner l'observation $\boldsymbol{x}_i$ au premier groupe (disons le groupe dénoté avec $y = 1$) si $h(\boldsymbol{x}_i) > 0$, et assigner au groupe -1 si $h(\boldsymbol{x}_i) < 0$. Ainsi, la construction de la fonction de perte zéro-un est liée au signe de la marge m_i de chaque observation $\boldsymbol{x}_i$. En effet, selon notre règle de classement définie ci-haut, nous avons que la marge m_i est positive si seulement si y_i et $h(\boldsymbol{x}_i)$ ont le même signe et elle est négative si y_i et $h(\boldsymbol{x}_i)$ ont deux signes différents. Ainsi, une marge $m_i < 0$ indique alors que l'observation $\boldsymbol{x}_i$ est mal-classée, tandis qu'une marge $m_i \geq 0$ indique que l'observation $\boldsymbol{x}_i$ est correctement classée. La fonction de perte zéro-un est définie ainsi comme suit Nguyen et Sanner (2013)

$$\ell_{0|1}(\boldsymbol{m}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left(m_i < 0\right),$$

avec $\mathbb{1}(t < 0) = 1$ si $t < 0$ et $\mathbb{1}(t < 0) = 0$ si $t \geq 0$. On peut noter que ce sont seulement les observations qui sont mal-classées qui contribuent à cette fonction de perte (c-à-d. les observations avec $m_i < 0$).

Puisque la marge $\boldsymbol{m}$ dépend des coefficients de $\boldsymbol{\beta}$ et de l'ordonnée à l'origine β_0 et que la fonction de perte zéro-un est discontinue et n'est pas dérivable, alors, résoudre un problème d'optimisation avec la fonction de perte $\ell_{0|1}(\boldsymbol{m})$ est un problème très difficile. En effet, les méthodes de la descente par gradient ou sous-gradient ne peuvent pas être appliquées pour minimiser le problème d'optimisation sous-jacent. D'où l'utilité d'utiliser d'autres fonctions de perte alternatives. D'autres fonctions de

pertes plus lisses comme, entre autres, la fonction de perte SVM ont été proposées dans la littérature pour estimer $\beta_0, \boldsymbol{\beta}$ et faciliter le classement binaire. La fonction de perte SVM est introduite à la section suivante.

1.1.2 La fonction de perte SVM

La fonction de perte SVM est définie par

$$\ell(\boldsymbol{m}) = \frac{1}{n}\sum_{i=1}^{n}[1 - m_i]_+, \tag{1.2}$$

avec $[1-t]_+ = \max(0, 1-t)$ et m_i est définie dans (1.1). Pour mieux comprendre cette fonction de perte, nous supposons également que les données sont linéairement séparables. Dans ce cas, il existe une infinité d'hyperplans qui séparent les données en deux classes 1 et -1. Ainsi, l'objectif de la méthode SVM est de chercher un hyperplan unique et optimal. Le problème d'optimisation sous-jaçant à la fonction de perte SVM permet l'obtention d'un hyperplan unique et optimal qui sépare les deux classes en maximisant la marge entre les observations et le dit hypérplan séparateur. Ainsi, nous pouvons redéfinir la règle de décision comme suit : nous affectons $\boldsymbol{x}_i$ à la classe 1 si $h(\boldsymbol{x}_i) > c$ et à la classe -1 si $h(\boldsymbol{x}_i) < -c$ où c est une constante positive. Nous pouvons reformuler cette règle comme suit : nous affectons $\boldsymbol{x}_i$ à la classe 1 si $h(\boldsymbol{x}_i) > 1$ et à la classe -1 si $h(\boldsymbol{x}_i) < -1$ car nous pouvons diviser les deux côtés par c. Ainsi, si la marge $m_i = y_i h(\boldsymbol{x}_i) > 1$, l'observation $\boldsymbol{x}_i$ est correctement classée et donc elle ne contribue pas dans la fonction de perte car $[1 - m_i]_+ = 0$. Lorsque $m_i < 1$, l'observation $\boldsymbol{x}_i$ est mal classée et sa contribution dans la fonction de perte SVM est égale à $1 - m_i$. La quantité $1 - m_i$ augmente pour les points qui s'éloignent de leur classe, car m_i pourrait prendre des valeurs beaucoup plus petites que 1, voire même des valeurs négatives, pour les points mals classés. Ainsi, ils sont pénalisés davantage dans la fonction de perte SVM. Finalement, les observations qui vérifient $m_i = y_i h(\boldsymbol{x}_i) = 1$ sont nommés des vecteurs supports, car ils sont utiles pour estimer l'hyperplan comme nous allons voir avec plus de détails à la section 1.2.

1.1.3 La fonction de perte Logistique

Dans le cadre de classement binaire, le modèle logistique consiste à modéliser $\boldsymbol{P}(y_i|\boldsymbol{x}_i)$. Ainsi, nous avons

$$p(\boldsymbol{x}_i) := \boldsymbol{P}(y_i = 1|\boldsymbol{x}_i) = \frac{1}{1 + \exp(-h(\boldsymbol{x}_i))}.$$

Nous avons également $\frac{y_i+1}{2} \in \{0,1\}$ est une variable de Bernouilli. La log-vraisemblance est donnée donc par

$$\ell(y_i, \boldsymbol{x}_i) = \frac{y_i+1}{2} \log p(\boldsymbol{x}_i) + \frac{1-y_i}{2} \log(1-p(\boldsymbol{x}_i)). \tag{1.3}$$

Nous pouvons remarquer que la fonction $\ell(y_i, \boldsymbol{x}_i)$ dans l'équation (1.3) satisfait l'équation suivante :

$$\ell(y_i, \boldsymbol{x}_i) = -\log((1+\exp(-y_i h(\boldsymbol{x}_i)))\,.$$

En effet, si $y_i = 1$, nous remplaçons $p(\boldsymbol{x}_i)$ par sa formule dans l'équation (1.3) pour obtenir

$$\ell(y_i, \boldsymbol{x}_i) = -\log\left(1+\exp(-h(\boldsymbol{x}_i))\right).$$

Si $y_i = -1$, nous obtenons

$$\ell(y_i, \boldsymbol{x}_i) = -\log((1+exp(h(\boldsymbol{x}_i)))\,.$$

Ainsi, nous pouvons combiner les deux cas pour obtenir

$$\ell(y_i, \boldsymbol{x}_i) = -\log\left(1+\exp(-y_i h(\boldsymbol{x}_i))\right) = \ell(m_i) = -\log\left(1+\exp(-m_i)\right).$$

Ainsi, nous définissons la fonction de perte empirique logistique $L(\boldsymbol{m})$ comme la moyenne sur tous les $i \in \{1,...,n\}$ de la log-vraisemblance négative, $-\ell(y_i, \boldsymbol{x}_i)$, définie dans l'équation (1.3). Alors, nous avons

$$L(\boldsymbol{m}) = -\frac{1}{n}\sum_{i=1}^{n} \ell(m_i). \tag{1.4}$$

Cette fonction est connue dans la littérature par le nom "la déviance binomiale" ou "l'entropie croisée" Hastie *et al.* (2009). Ainsi, la déviance binomiale est définie par l'équation suivante

$$D(m) = \log(1+\exp(-m)), \tag{1.5}$$

En effet, dans le cas binaire, la fonction de perte logistique et la déviance binomiale sont les mêmes Hastie *et al.* (2009). Notez que si la marge $m \to +\infty$, alors nous obtenons $D(m) = 0$, ce qui veut dire que l'observation $\boldsymbol{x}_i$ est correctement classée. Si $m \leq 0$, alors $D(m) \neq 0$. Autrement dit, encore une fois, si $m_i > 0$, la contribution de la ième observation dans la fonction de perte est moins importante qu'une observation avec $m_i < 0$.

1.1.4 La fonction de perte Huber

La fonction Huber Rosset et Zhu (2007), est une fonction de perte qui approxime la fonction de perte SVM donnée dans l'équation (1.2). La fonction de perte Huber est définie par l'équation suivante :

$$\phi_c(m_i) = \begin{cases} 0, & \text{si } m_i \geq 1 \\ \frac{(1-m_i)^2}{2\delta}, & \text{si } 1-\delta < m_i \leq 1 \\ 1 - m_i - \frac{\delta}{2}, & \text{si } m_i \leq 1-\delta, \end{cases} \tag{1.6}$$

où $\delta > 0$ est un paramètre de lissage et $i = 1, \ldots, n$. Le graphique suivant décrit le comportement de toutes les fonctions de pertes étudiées dans cette section.

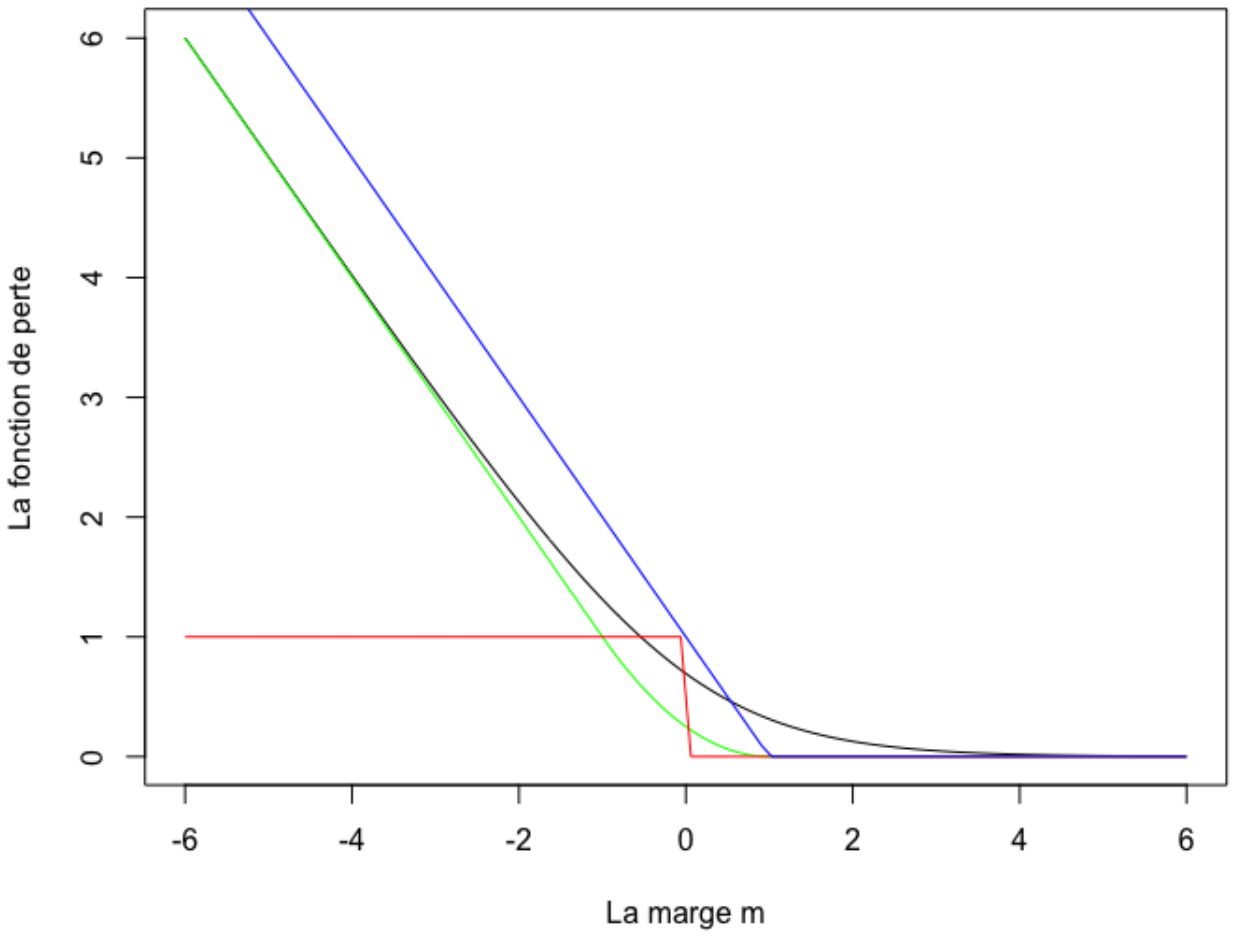

FIGURE 1.1 La fonction de perte zéro-un (rouge), la fonction Huber (vert) avec $\delta = 2$, la déviance binomiale $D(m)$ (noir) et la fonction de perte SVM (blue).

Nous pouvons constater que la fonction de perte SVM est un majorant pour la fonction de perte zéro-un. Cependant, les fonctions de pertes logistique et Huber ne le sont pas. Toutes les fonctions de pertes ont des queues à droite similaires. En effet, quand $m \geq 1$, toutes les fonctions de pertes donnent un zéro comme pénalité pour tous les points bien classés. En revanche, les points mals classés ou eloignés de plus sont pénalisés linéairement. La fonction de perte BernSVM proposée dans le troisième chapitre de la thèse est un majorant de la fonction de perte zéro-un qui a des queues similaires aux queues de la fonction de perte SVM, et c'est une fonction différentiable partout. Ceci

ce n'est pas le cas pour la fonction de perte SVM qui n'est pas différentiable en 1.

1.2 La méthode SVM

La méthode SVM est une technique très populaire de classification, connue sous le nom de séparation à vaste marge. Nous utilisons la méthode SVM pour résoudre des problèmes de classification (ou de discrimination).

Supposons que nous avons un ensemble de données d'entrainement $\{(\boldsymbol{x}_i, y_i), i = 1, \ldots, n\}$, où $\boldsymbol{x}_i \in \mathbb{R}^p$ avec p prédicteurs et $y_i \in \{-1, 1\}$. Nous redéfinissons l'hyperplan $H : \{\boldsymbol{x} : h(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\beta} + \beta_0 = 0\}$. Nous avons vu que la décision de classement est définie par $sign[h(\boldsymbol{x})]$. Nous supposons également que les données sont linéairement séparables, cela veut dire que nous pouvons séparer les deux classes par un hyperplan. Autrement dit, nous pouvons trouver une fonction $h(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\beta} + \beta_0$ avec une marge positive, $m_i = y_i h(\boldsymbol{x}_i) > 0$, pour tout $i = 1, ..., n$. Une infinité d'hyperplans satisfaisant ce critère. Ainsi, l'objectif de la méthode SVM est de trouver l'hyperplan séparateur optimal avec la plus grande marge, M, séparant les données pour les classes -1 et 1. Le graphique suivant illustre cette situation dans le cas $p = 2$. Dans ce cas, l'hyperplan est une droite.

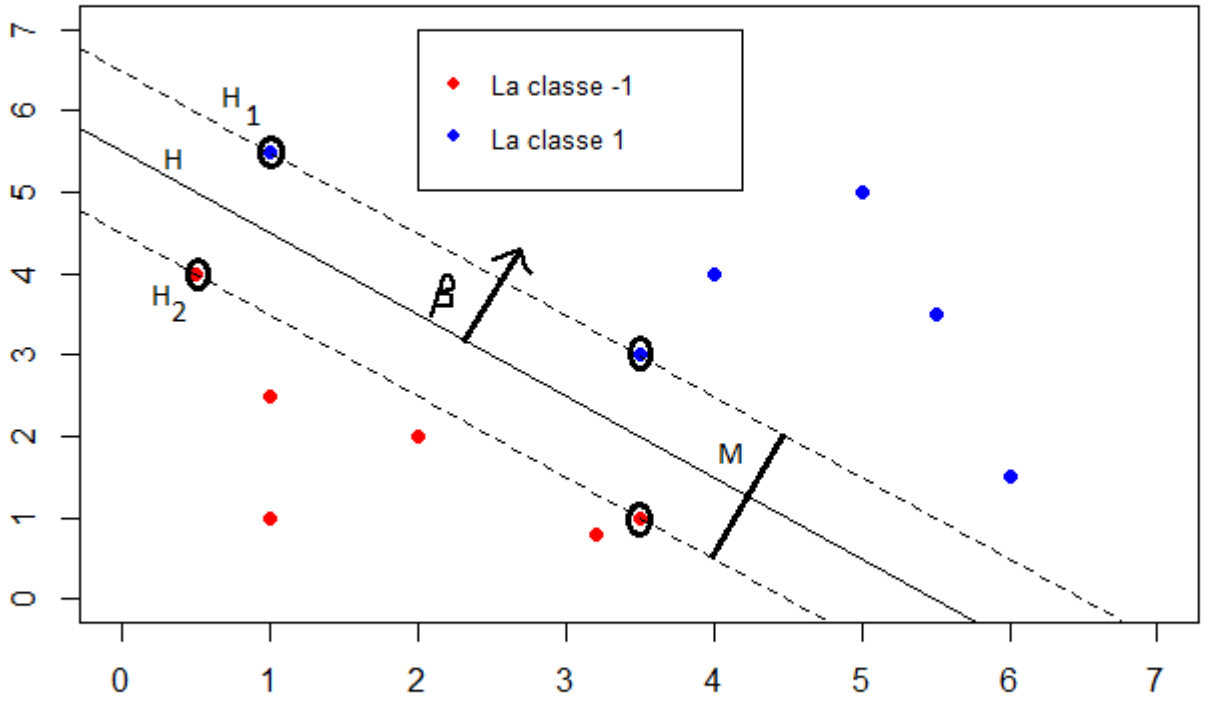

FIGURE 1.2 L'hyperplan séparant deux classes qui sont linéairement séparables.

L'équation de l'hyperplan H_1 est donnée par l'équation suivante

$$\boldsymbol{x}^\top \boldsymbol{\beta} + \beta_0 = 1,$$

tandis que l'équation de l'hyperplan H_2 est donnée par l'équation suivante

$$\boldsymbol{x}^\top \boldsymbol{\beta} + \beta_0 = -1.$$

Nous pouvons observer dans la Figure 1.2 que les points de la classe 1 vérifient l'inégalité

$$\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0 \geq 1,$$

et les points de la classe 2 vérifient l'inégalité

$$\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0 \leq -1.$$

Nous pouvons combiner les deux inéquations pour aboutir à l'inégalité suivante :

$$y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1 \quad \forall i. \tag{1.7}$$

Les points encerclés sont des vecteurs supports. Ainsi, la méthode SVM consiste à résoudre le problème d'optimisation suivant :

$$\max_{\boldsymbol{\beta},\beta_0} M \quad \text{sous contrainte} \quad y_i(\boldsymbol{x}_{i.}^\top \boldsymbol{\beta} + \beta_0) \geq 1, i = 1, ..., n. \tag{1.8}$$

La marge M dans la Figure 1.2 est la distance (orthogonale) entre les deux hyperplans H_1 et H_2. Nous avons, $d(\boldsymbol{x} \in H_k, H) = \frac{|\boldsymbol{x}^\top \boldsymbol{\beta} + \beta_0|}{||\boldsymbol{\beta}||} = \frac{1}{||\boldsymbol{\beta}||}$, pour $k = 1, 2$. Ainsi, la marge M est donnée par la formule suivante

$$M = \frac{2}{||\boldsymbol{\beta}||}.$$

Nous pouvons remarquer que maximiser la marge M équivaut à maximiser $2/||\boldsymbol{\beta}||$, ce qui implique la minimisation de $||\boldsymbol{\beta}||$. Il vaut mieux minimiser le carré $||\boldsymbol{\beta}||_2^2$ afin d'éviter de dériver une fonction racine carrée. Ainsi, nous pouvons réécrire le problème d'optimisation dans (1.8) sous la forme suivante

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{2}||\boldsymbol{\beta}||^2 \quad \text{sous contrainte} \quad y_i(\boldsymbol{x}_{i.}^\top \boldsymbol{\beta} + \beta_0) \geq 1, i = 1, ..., n. \tag{1.9}$$

Pour les données qui ne sont pas linéairement séparables, nous introduisons des variables positives $(\xi_1, ..., \xi_n)$ et les associer aux pairs $(\boldsymbol{x}_{i.}, y_i)$, $i = 1, ..., n$. Ces écarts ξ_i sont définis de la façon suivante : l'observation $(\boldsymbol{x}_i, y_i)$ est bien classée si $y_i h(\boldsymbol{x}_i) \geq 1$. D'où, $\xi_i = 0$; l'observation $(\boldsymbol{x}_i, y_i)$ est mal classée si $y_i h(\boldsymbol{x}_i) < 1$ et dans ce cas, on a $\xi_i = 1 - y_i h(\boldsymbol{x}_i)$. Dans ce cas, la contrainte dans l'équation (1.9) devient $y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq (1 - \xi_i), i = 1, ..., n$, sachant que $\sum_{i=1}^n \xi_i \leq C$, où $C > 0$. Les écarts ξ_i nous indiquent de combien une prédiction pour une observation $\boldsymbol{x}_i$ est mal positionnée par rapport à sa marge. Le problème de minimisation suivant est une forme générale du problème donné par l'équation (1.9)

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{2}||\boldsymbol{\beta}||^2 \quad \text{sous contrainte} \quad \begin{cases} y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i \quad \forall i = 1, ..., n \\ \xi_i \geq 0, \quad \sum_{i=1}^n \xi_i \leq C. \end{cases} \tag{1.10}$$

Le problème (1.10) peut s'écrire sous la forme lagrangienne comme suit

$$\begin{aligned} &\min_{\boldsymbol{\beta},\beta_0} \tfrac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^n \xi_i \\ &\text{sous contrainte} \quad \xi_i \geq 0, y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i \quad \forall i. \end{aligned} \tag{1.11}$$

Ainsi, le problème (1.11) peut être résolu en utilisant les multiplicateurs de Lagrange. En effet, le lagrangien est donné par

$$L = \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(\mathbf{x}_{i.}^\top\boldsymbol{\beta} + \beta_0) - (1-\xi_i)] - \sum_{i=1}^{n}\mu_i\xi_i.$$

Nous allons minimiser L par rapport à $\boldsymbol{\beta}$, β_0 et $\boldsymbol{\xi}$. D'abord, la dérivée partielle de L par rapport à $\boldsymbol{\beta}$ est donnée par

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} + 0 - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i.}^\top - 0,$$

ce qui implique que

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i.}^\top.$$

En égalisant à zéro, on obtient

$$\boldsymbol{\beta} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i.}^\top.$$

Ensuite, on calcule la dérivée partielle de L par rapport à β_0 comme suit

$$\frac{\partial L}{\partial \beta_0} = 0 + 0 - \sum_{i=1}^{n}\alpha_i y_i - 0.$$

En posant cette dérivée égal à zéro, on obtient $\sum_{i=1}^{n}\alpha_i y_i = 0$. Finalement, la dérivée partielle de L par rapport à $\boldsymbol{\xi}$ est donnée par

$$\frac{\partial L}{\partial \xi_i} = 0 + C - \alpha_i - \mu_i.$$

En égalisant cette dérivée égal à zéro, on trouve $\alpha_i = C - \mu_i$, $\forall i$. Puis, on remplace les quantités trouvées par leurs valeurs dans l'expression de L dans l'expression ci-haut. Alors, on obtient

$$\tilde{L} = \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i.}^\top(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i.}) - \sum_{i=1}^{n}\alpha_i y_i \beta_0 + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}\mu_i\xi_i.$$

Après simplification, on obtient

$$\tilde{L} = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j.$$

On minimise $\tilde{L}$ afin d'obtenir $\hat{\alpha}_i$, ensuite, on utilise cette dernière pour calculer $\hat{\boldsymbol{\beta}}$ comme suit

$$\hat{\boldsymbol{\beta}} = \sum_{i=1}^{n}\hat{\boldsymbol{\alpha}}_i y_i \mathbf{x}_{i.}.$$

Ainsi, nous pouvons remarquer que seulement les vecteurs supports sont considérés dans le calcul des coefficients du modèle.

1.2.1 La méthode SVM vue comme une méthode de pénalisation

Si nous prenons $\lambda = \frac{1}{nC}$, le problème d'optimisation défini dans l'équation (1.11) peut s'écrire sous la forme équivalente suivante

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{n}\sum_{i=1}^{n}[1 - y_i(\beta_0 + \boldsymbol{x}_{i.}^\top \boldsymbol{\beta})]_+ + \frac{\lambda}{2}||\boldsymbol{\beta}||^2. \tag{1.12}$$

En effet, les contraintes dans l'équation (1.11) :

$$\xi_i \geq 0, \quad y_i(\beta_0 + \boldsymbol{x}_{i.}^\top \boldsymbol{\beta}) \geq 1 - \xi_i \forall i,$$

peuvent s'écrire d'une manière équivalente comme suit :

$$\xi_i \geq [1 - y_i(\beta_0 + \boldsymbol{x}_{i.}^\top \boldsymbol{\beta})]_+ \forall i,$$

oú

$$[1-t]_+ = \begin{cases} 0, & \text{si } t \geq 1 \\ 1-t, & \text{si } t \leq 1. \end{cases}$$

Ainsi, minimiser (1.11) est équivaut à minimiser (1.12). L'écriture de la solution de la méthode SVM définie à (1.12) sous forme d'une fonction de perte pénalisé ouvert la porte à plusieurs extensions de la méthode SVM avec des pénalités existentes. Nous pouvons ainsi écrire le problème (1.12) sous la forme générale

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{n}\sum_{i=1}^{n}[1 - y_i(\beta_0 + \boldsymbol{x}_{i}^\top \boldsymbol{\beta})]_+ + P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}), \tag{1.13}$$

où λ_1, λ_2 sont deux paramètres de régularisation positifs et $P_{\lambda_1,\lambda_2}(\boldsymbol{\beta})$ est une pénalité qui peut prendre différentes formes :

- Ridge (SVM standard)

$$P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) = \frac{\lambda_2}{2}||\boldsymbol{\beta}||_2^2;$$

- Lasso Bradley et Mangasarian (1998)

$$P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) = \lambda_1||\boldsymbol{\beta}||_1;$$

- Élastique Net (EN) Wang *et al.* (2006)

$$P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) = \lambda_1||\boldsymbol{\beta}||_1 + \frac{\lambda_2}{2}||\boldsymbol{\beta}||_2^2;$$

- SCAD Fan et Li (2001)

$$P_{\lambda_1,\lambda_2}(|\beta_j|) = \lambda_1|\beta_j|\mathbf{I}_{|\beta_j|\leq\lambda_1} - \frac{|\beta_j|^2 - 2a\lambda_1|\beta_j| + \lambda_1^2}{2(a-1)}\mathbf{I}_{\lambda_1<|\beta_j|\leq a\lambda_1} + \frac{(a+1)\lambda_1^2}{2}\mathbf{I}_{|\beta_j|>a\lambda_1}$$

  oú $a > 2$;

- MCP Zhang (2010)

$$P_{\lambda_1,\lambda_2}(|\beta_j|) = \lambda_1(|\beta_j| - \frac{|\beta_j|^2}{2\lambda_1 a})\mathbf{I}_{|\beta_j|<\lambda_1 a} + \frac{\lambda_1^2 a}{2}\mathbf{I}_{|\beta_j|\geq\lambda_1 a},$$

  où $a > 1$.

1.2.2 La méthode SVM avec un regroupement inconnu des variables explicatives

Dans le chapitre 2, afin de traiter des données avec un regroupement inconnu des variables explicatives, nous avons construit une nouvelle pénalité CCN [de l'anglais cluster correlation-network] qui tient compte de cette structure dans les données pour régulariser la méthode SVM. La résolution du problème résultant fait intervenir la méthode de regroupement K-means. K-means est une méthode d'apprentissage automatique non supervisé que nous utilisons comme outil pour partitionner un ensemble d'observations ou une matrice de données en K ensembles. Son principe est basé sur le classement des observations similaires dans un même groupe.

Notation : Soit $\boldsymbol{L}_1, \boldsymbol{L}_2, ..., \boldsymbol{L}_n$, les vecteurs colonnes d'une matrice de données, $\boldsymbol{L}(n \times p)$, avec $\boldsymbol{L}_j \in \mathbb{R}^p$. L'objectif est de partitionner ces vecteurs en K groupes, $C_1, ..., C_K$, avec K est un nombre fixé à priori.

Soit $\boldsymbol{\mu}_k \in \mathbb{R}^n$ le vecteur moyen des vecteurs classés dans le groupe C_k. La variation totale inter-groupes est définie par

$$\sigma^2 = \sum_{k=1}^{K} \sum_{\boldsymbol{L}_j \in C_k} ||\boldsymbol{L}_j - \boldsymbol{\mu}_k||^2.$$

L'algorithme K-means consiste à trouver les groupes $C_1, ..., C_K$, qui minimisent la quantité σ^2. Les détails de cet algorithme sont donnés dans Algorithme 1.

Algorithm 1 L'algorithme K-means.

1. On initialise la matrice des centres (les vecteurs moyens des K groupes), $\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K$, aléatoirement ;
2. On répète les deux étapes suivantes jusqu'à convergence :
   — Pour chaque $j \in \{1, ..., p\}$, on pose

$$c_j = argmin_{k \in \{1,...,K\}} ||\boldsymbol{L}_j - \boldsymbol{\mu}_k||^2.$$

   — Pour chaque k, nous faisons la mise à jour des $\boldsymbol{\mu}_k$

$$\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{j \in C_K} L_j,$$

   avec $C_k = \{j | c_j = k\}$.

Les $\boldsymbol{\mu}_k, k = 1, ..., K$, désignent les centres des groupes $C_1, ..., C_K$. L'indice c_j désigne le groupe C_k où le vecteur $\boldsymbol{L}_j$ est assigné, et $|C_k|$ désigne le nombre d'élements dans le groupe C_k. Notez que dans cette section, nous nous sommes intéressés à regrouper les colonnes de la matrice $\boldsymbol{L}$. On peut procéder de façon simulaire si on s'intéresse à regrouper les lignes de $\boldsymbol{L}$.

1.3 Problème de non-différentiabilité de la fonction de perte SVM

Nous avons souligné que la fonction de perte SVM n'est pas différentiable partout. En effet, elle n'est pas dérivable pour $t = 1$. Ainsi, nous ne pouvons pas résoudre un problème pénalisé avec la fonction SVM en utilisant une descente par gradient. Pour régler ce problème de dérivabilité, une approximation lisse de la fonction SVM est souhaitable. Ainsi, dans notre premier travail, nous avons approximé la fonction SVM par la fonction Huber définie par l'équation (1.6). Cette fonction de perte est dérivable partout, donc nous pouvons appliquer la descente par gradient pour résoudre le problème d'optimisation SVM pénalisé. Afin de dériver des propriétés non asymptotiques des estimateurs des coefficients $\boldsymbol{\beta}$, nous avons besoin également de la dérivée seconde. La fonction Huber ne possède pas cette propriété car elle n'est pas deux fois dérivable. Cependant, dans notre deuxième travail (chapitre 3), nous avons proposé une approximation lisse de la fonction SVM. Cette fonction de perte est nommée BernSVM [de l'anglais Bernstein SVM]. C'est une fonction que nous avons construite en utilisant des techniques d'approximation d'une fonction par les polynômes de

Bernstein. La section suivante décrit un aperçu général sur l'approximation d'une fonction par les polynômes de Bernstein.

1.4 Approximation d'une fonction en utilisant les splines cubiques et les polynômes de Bernstein

Dans cette section, nous allons présenter une méthode d'approximation d'une fonction continue $f(x)$, définie sur un intervalle $[a, b]$, en utilisant les splines cubiques et les polynômes de Bernstein Lorentz (2013). D'abord, nous allons commencer par définir les polynômes de Bernstein. Ensuite, nous allons définir les splines cubiques, et finalement présenter la méthode à suivre pour approximer la fonction $f(x)$. Nous allons donner un exemple d'application en utilisant la déviance binomiale définie par (1.5).

1.4.1 Les polynômes de Bernstein

Les membres d'une base de Bernstein de degré $m > 0$ sont les polynômes définis par

$$b_{k,m} = \begin{pmatrix} m \\ k \end{pmatrix} x^k (1-x)^{m-k}, \ x \in \ [0, \ 1],$$

avec $0 \leq k \leq m$. Nous pouvons écrire un polynôme $P(x)$ dans une base de Bernstein sous la forme suivante :

$$P(x) = \sum_{k=0}^{m} c_k b_{k,m}(x), \quad x \in \ [0, \ 1],$$

avec les coefficients $c_k, k = 0, ..., m$, sont déterminés avec des contraintes imposées sur le polynôme $P(x)$.

1.4.2 Les splines cubiques

Les splines cubiques sont des polynômes de degré au plus 3. Ils sont utilisés pour lisser une fonction $f(x)$ sur un intervalle $[a, b]$. En effet, supposons que nous avons un ensemble de points ordonnés en abscisses $(x_0, y_0), ..., (x_n, y_n)$, où $x_0 = a$ et $x_n = b$, appartenant au support de la fonction $f(x)$. L'objectif est de construire une fonction $S(x)$ qui approxime $f(x)$, deux fois dérivable et continue

par morceaux, de la façon suivante

$$S(x) = \begin{cases} P_1(x) \quad x \in [x_0,\ x_1]; \\ . \\ . \\ . \\ P_n(x) \quad x \in [x_{n-1},\ x_n]. \end{cases}$$

Les polynômes $P_i(x)$ sont de degré inférieur ou égale à 3, pour $i = 1, ..., n$, et satisfont les conditions suivantes :

- $P_1(x_0) = y_1$ et $P_n(x_n) = y_n$;
- $P_{i+1}(x_i) = P_{i+2}(x_i)$ pour chaque $i = 1, ..., n-3$;
- $P'_{i+1}(x_i) = P'_{i+2}(x_i)$ pour chaque $i = 1, ..., n-3$;
- $P''_{i+1}(x_i) = P''_{i+2}(x_i)$ pour chaque $i = 1, ..., n-3$;
- $S(x_i) = y_i$ pour chaque $i = 0, ..., n$.

Remarque : Pour définir la fonction $S(x)$ de façon unique, nous allons ajouter une condition supplémentaire. Entre autres, la condition $S''(a) = S''(b) = 0$, dans ce cas la fonction $S(x)$ est appelée une spline naturelle. Nous pouvons également supposer $S'(a) = f'(a)$ et $S'(b) = f'(b)$.

1.4.3 Exemples d'application

- **Exemple 1 :**

  Nous considérons la fonction

  $$f(t) = |t - 1|.$$

  Nous constatons que la fonction $f(t)$ est continue et convexe, mais n'est pas dérivable partout car elle n'est pas dérivable en 1. Ainsi, nous allons approximer la fonction $f(x)$ par une fonction, $g(x)$, continue et deux fois dérivable. Nous allons établir un paramètre de lissage $\delta > 0$, et utiliser les polynômes de Bernstein pour définir la fonction suivante :

  $$S_\delta(t) = \begin{cases} 1 - t\ , & \text{if } t - 1\ < -\delta \\ t - 1\ , & \text{if } t - 1\ > \delta \\ g_\delta(t)\ , & \text{if } |t - 1|\ \leq \delta, \end{cases}$$

  avec g_δ est un polynôme et que $S_\delta(.)$ est de classe C^2. En particulier, nous voulons que $S_\delta(t) \to f(t)$ si $\delta \to 0$, pour tout $t \in \mathbb{R}$. Le polynôme $g_\delta(.)$ vérifie les conditions suivantes :

C1. $g_\delta(1-\delta) = \delta$, $g_\delta^{'}(1-\delta) = -1$, and $g_\delta^{''}(1-\delta) = 0$;

C2. $g_\delta(1+\delta) = \delta$, $g_\delta^{'}(1+\delta) = 1$, and $g_\delta^{''}(1+\delta) = 0$;

C3. $g_\delta^{''} \geq 0$.

Nous allons prendre en compte la transformation affine $t \mapsto (t-1+\delta)/2\delta$, afin de transformer l'intervalle $[1-\delta,\ 1+\delta]$ en $[0,\ 1]$, et supposer

$$q_\delta(x) = g_\delta(2\delta x + 1 - \delta), \quad x \in\ [0,\ 1].$$

Les conditions C1, C2 et C3, s'expriment en termes de $q_\delta(.)$

C1'. $q_\delta(0) = \delta$, $q_\delta^{'}(0) = -2\delta$, and $q_\delta^{''}(0) = 0$;

C2'. $q_\delta(1) = \delta$, $q_\delta^{'}(1) = 2\delta$, and $q_\delta^{''}(1) = 0$;

C3'. $q_\delta^{''} \geq 0$.

Notez que C1' et C2' implique que le polynôme $g_\delta(.)$ sera de degré 4 car nous ne pouvons pas trouvé un polynôme de degré 3 qui satisfait aux conditions ci-haut. En effet, si ce polynôme existe, sa dérivée seconde va être une fonction affine qui s'annule dans les extrémités et donc avec la continuité va être nul partout. Après avoir calculé les coefficients c_k, $k = 0, 1, \ldots, 4$, nous obtenons

$$q_\delta(x) = \delta(1-x)^4 + 2\delta x(1-x)^3 + 2\delta x^3(1-x) + \delta x^4,$$

et pour trouver $g_\delta(.)$, il suffit de prendre

$$g_\delta(t) = q_\delta((t-1+\delta)/2\delta).$$

La Figure 1.3 décrit le comportement de la fonction de lissage $S_\delta(t)$ pour les différentes valeurs de $\delta \in \{0.3, 1, 2, 4\}$.

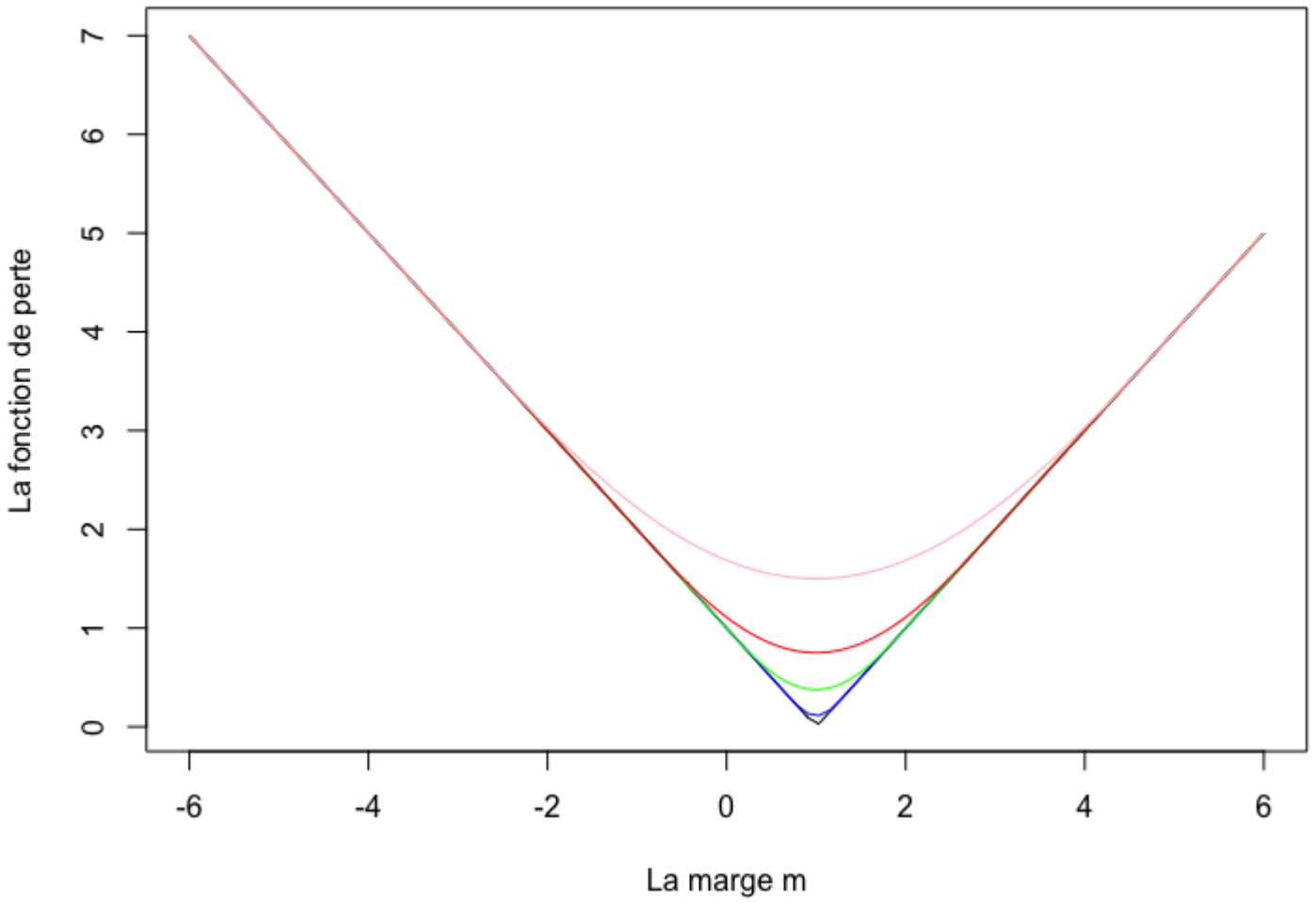

FIGURE 1.3 La fonction $f(m)$ en noir et son approximation $S_\delta(m)$ pour $\delta = 0.3$ (bleu), $\delta = 1$ (vert), $\delta = 2$ (rouge) et pour $\delta = 4$ (rose).

- **Exemple 2 :**

  Dans cet exemple, nous avons illustré le lissage de la fonction Déviance Binomiale définie par l'équation (1.5) sur l'intervalle $[-3, 3]$ en utilisant les polynômes de Bernstein. En effet, $D(x) = \log(1 + e^{-x})$, donc $D(-3) = 3.05$ et $D(3) = 0.05$. La dérivée première de $D(.)$ est donnée par $D^{'}(x) = \frac{-1}{1+e^{x}}$, ainsi $D^{'}(-3) = -0.95$ et $D^{'}(3) = -0.05$. Nous cherchons la fonction $S(x)$ qui lisse la fonction $D(x)$ sur l'intervalle $[-3, 3]$. La fonction $S(x)$ vérifie les conditions suivantes : $S(-3) = 3.05; S(3) = 0.05; S^{'}(-3) = -0.95$ et $S^{'}(3) = -0.05$. Pour

pouvoir utiliser la base de Bernstein, nous allons faire la transformation linéaire suivante $x \to \frac{x+3}{6}$, ce qui transforme $[-3, 3]$ en $[0, 1]$.

Soit $q(x) = S(6x-3)$ pour $x \in [0, 1]$. La fonction $q(x)$ s'écrit dans la base de Bernstein de la façon suivante $q(x) = \sum_{k=0}^{3} c_k b_{k,3}$ ou $b_{0,3} = (1-x)^3, b_{1,3} = 3x(1-x)^2, b_{2,3} = 3x^2(1-x)^2$ et $b_{3,3} = x^3$. Ainsi, nous avons

$$q(x) = c_0(1-x)^3 + 3c_1x(1-x)^2 + 3c_2x^2(1-x) + c_3x^3.$$

Pour déterminer les coefficients $c_k, k = 0, 1, 2, 3$, nous allons calculer d'abord $q(0), q(1), q^{'}(0)$ et $q^{'}(1)$. En effet, $q(0) = S(-3) = 3.05$, $q(1) = S(3) = 0.05$, $q^{'}(0) = 6S^{'}(-3) = -5.7$ et $q^{'}(1) = 6S^{'}(3) = -0.3$. Nous avons quatre coefficients et quatre équations. Après la résolution de ces équations, nous obtenons

$$q(x) = 3.05(1-x)^3 + 3.45x(1-x)^2 + 0.15x^2(1-x) + 0.05x^3.$$

En conséquence, la fonction de lissage est donnée par

$$S(x) = q\left(\frac{x+3}{6}\right) = [3.05(3-x)^3 + 3.45(3-x)^2(x+3) + 0.15(x+3)^2(3-x) + 0.05(x+3)^3]/6^3.$$

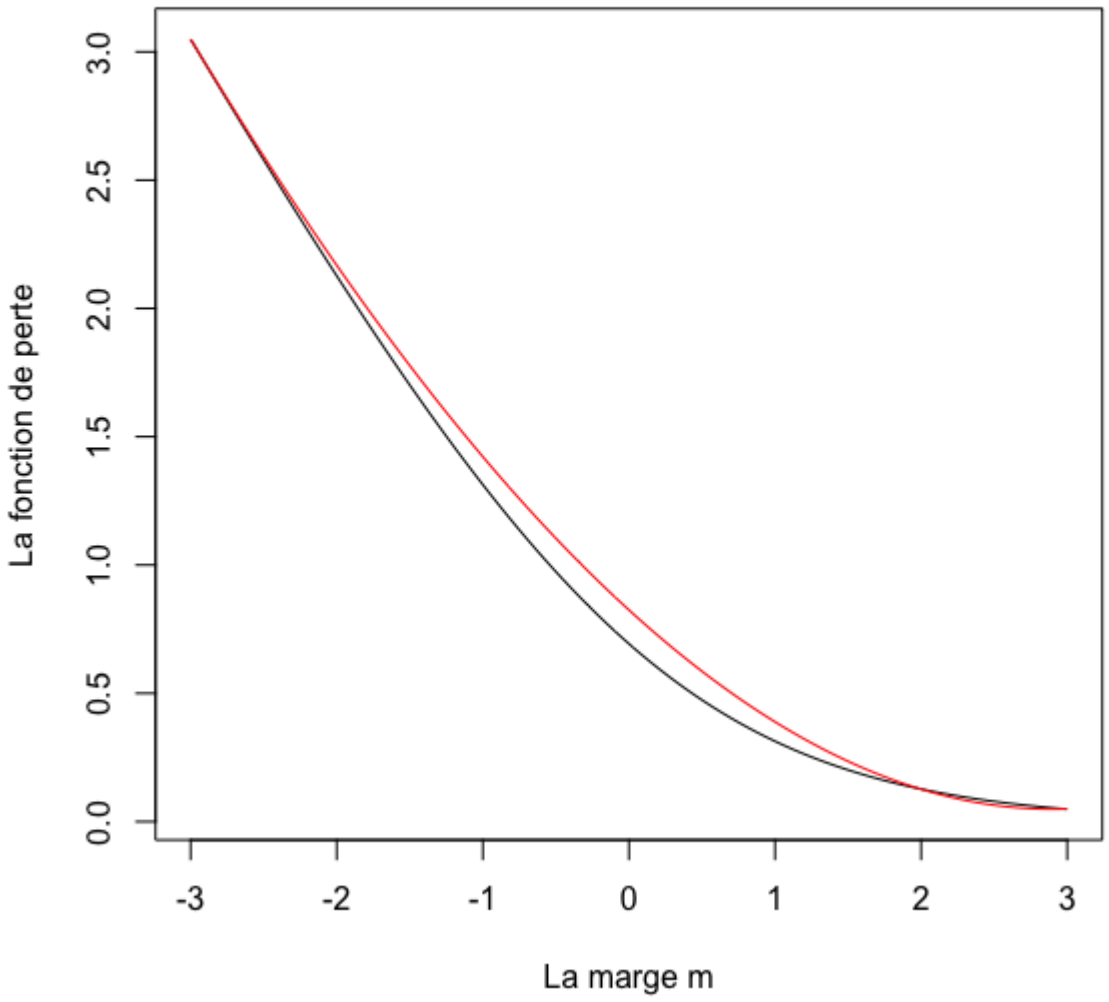

FIGURE 1.4 La fonction $D(m)$ en noir et son approximation $S(m)$ en rouge.

1.5 La descente par coordonnée et le principe Majoration-Minimisation

Dans cette section, nous allons présenter un aperçu général sur l'utilité des deux algorithmes DC et MM. Nous considérons la fonction objective à plusieurs variables $F(\boldsymbol{\beta})$ où $\boldsymbol{\beta} \in \mathbb{R}^p$ est un vecteur inconnu. Nous supposons que $F(.)$ est convexe et dérivable. La méthode DC Tseng (2001) est une approche très populaire et très utile pour résoudre des problèmes de minimisation lorsque la fonction $F(\boldsymbol{\beta})$ a de belles propriètés (c.à.d, dérivable et convexe). En effet, à chaque itération, nous minimisons

$F(\boldsymbol{\beta})$ par rapport à une coordonnée, en gardant les autres coordonnées fixes. S'il existe un point $\boldsymbol{\beta}$, tel que $F(\boldsymbol{\beta})$ est minimisée en chaque coordonnée β_j, alors $F(\boldsymbol{\beta})$ admet un minimum global Tseng (2001). L'algorithme suivant nous montre comment trouver ce minimum global. L'algorithme procède de la façon suivante :

L'algorithme DC

* On commence par $\boldsymbol{\beta}^{(0)}$.
* On répète pour tout $k = 1, 2, 3, ...$

$$\beta_1^{(k)} \in \arg\min_{\beta_1} F(\beta_1, \beta_2^{(k-1)}, \beta_3^{(k-1)}, .., \beta_p^{(k-1)})$$

$$\beta_2^{(k)} \in \arg\min_{\beta_2} F(\beta_1^{(k)}, \beta_2, \beta_3^{(k-1)}, .., \beta_p^{(k-1)})$$

$$\beta_3^{(k)} \in \arg\min_{\beta_3} F(\beta_1^{(k)}, \beta_2^{(k)}, \beta_3, .., \beta_p^{(k-1)})$$

$$.$$

$$.$$

$$.$$

$$\beta_p^{(k)} \in \arg\min_{\beta_p} F(\beta_1^{(k)}, \beta_2^{(k)}, \beta_3^{(k)}, .., \beta_p)$$

jusqu'à convergence, c.-à-d. jusqu'à

$$\beta_j^{(k)} \simeq \beta_j^{(k-1)}, \quad j = 1, ..., p.$$

Le principe MM Hunter et Lange (2004) consiste à majorer la fonction $F(\beta_j)$ par une autre fonction assez lisse et facile à minimiser. La propriété de la descente garantie que la minimisation de la nouvelle approximation est équivalente à la minimisation de la fonction $F(\beta_j)$. Ces deux techniques d'optimisation sont illustrés de façon approfondies dans les trois contributions (chapitres 2,3 et 4) de la thèse.

CHAPITRE 2
THE CLUSTER CORRELATION-NETWORK SUPPORT VECTOR MACHINE FOR HIGH-DIMENSIONAL BINARY CLASSIFICATION

En grande dimension, la présence de données avec des covariables fortement corrélées est très fréquent. Ainsi, dans le cadre du classement avec une variable réponse binaire, identifier les groupes de prédicteurs associés à cette variable s'avère très difficile. Par conséquent, nous proposons CCNSVM, une nouvelle approche qui estime simultanément les groupes de prédicteurs importants pour la classification et les coefficients des paramètres de la méthode SVM pénalisée. La pénalité CCN est une combinaison des vecteurs de la matrice des chevauchements topologiques [TOM, de l'anglais Topoligical Overlap Matrix]. Cette dernière mesure la connectivité entre les prédicteurs. L'algorithme proposé pour résoudre le problème CCNSVM se fait en deux étapes. D'abord, nous utilisons K-means pour regrouper les prédicteurs en K groupes. Ensuite, nous utilisons les groupes estimés pour calculer les coefficients des paramètres de la méthode CCNSVM pénalisée en combinant le principe de maximisation-minimisation et la descente par coordonnée. Cette méthode favorise la sparsité et le regroupement de prédicteurs importants pour la variable réponse. L'exploration de données simulées et l'analyse de données réelles illustrent l'importance et l'éfficacité de la méthode CCNSVM.

Abstract

Identifying homogeneous subsets of predictors associated with binary classification response can be challenging in the presence of high-dimensional data with highly correlated variables. We propose a new method called Cluster Correlation-Network SVM, CCNSVM for short, that simultaneously estimates clusters of predictors that are relevant for classification and coefficients of penalized SVM. The new CCN penalty is a function of the well-known Topological Overlap Matrix (TOM) whose general term measures the strength of connectivity between two predictors. This is achieved via an implementation of an efficient algorithm which alternates between two steps until convergence. Regarding the first step, a search of clustering the predictors into K groups is first obtained ; then step two consists of optimizing a penalized SVM function with fixed groups from the first step using Majorization-Minimization tricks together with a coordinate descent algorithm. This combining of clustering and sparsity into a single procedure provides additional insights into the power of exploring dimension reduction structure in high-dimensional binary classification. Simulation studies are considered to compare the performance of our procedure to its competitors. A practical application of CCNSVM on bisulfite sequencing DNA methylation data illustrates its good behaviour.

Keywords: Support Vector Machine, Classification, Clustering, Variables Selection, Shrinkage.

2.1 Introduction

The support vector machine (SVM) was developed around 1990s by the computer science community (cf. Vapnik (1999) and Cortes et Vapnik (1995)). It is a powerful tool widely used for binary classification and prediction problems. Its motivation comes from geometric considerations and it can be considered as a generalization of an intuitive and elegant classifier called the *maximal margin classifier* (also known as the *optimal separating hyperplane*) (cf. Gareth *et al.* (2013), ch. 9). The latter seeks a hyperplan that separates two classes of data points by the largest margins (i. e. minimal distances of observations to a hyperplan). Next, the decision rule consists of classifying a test observation based on which side of the maximal margin hyperplane it lies. This classification rule depends only on a subset of observations, called *support vectors*, which lie along the lines indicating the width of the margin. However, it is adapted only to a separating hyperplan which does not always exist. *The support vector classifier* (SVC) is the generalization of the maximal margin classifier to the non-separable case (see Section 2) and it will perform well if the boundary between the two classes is linear. Finally, the name SVM normally corresponds to an extension of the SVC that results from enlarging the feature space using some kernel tricks. Nevertheless, we keep the SVM denomination for SVC in the following.

In this paper we focus on binary classification with high-dimensional data where the number of variables p is much larger than the sample size n. In this setting, the sparsity condition assumes that only a few number of predictors will affect the classification rule. Therefore, the classical SVM rule is difficult to interpret, since all predictors are used for the construction of the classification rule. To overcome this problem, recently, interest has focused on an alternative class of sparse SVM approaches which implement both variable selection and coefficient shrinkage in a single procedure. Some of them which are widely used in practice are : SVM with the ℓ_1 penalty (ℓ_1SVM ; Bradley et Mangasarian (1998) ; Zhu *et al.* (2003), SVM with elastic net penalty (EnetSVM ; Wang *et al.* (2006), SVM with the Scad penalty (ScadSVM) and SVM with the combination of SCAD and ℓ_2 penalty (SnetSVM), Becker *et al.* (2011). However, ℓ_1SVM and ScadSVM do not take into account the correlation between predictors and tend to select only one predictor from a group in the presence of correlated predictors. On the other hand, EnetSVM and SnetSVM select correlated relevant variables and produce parsimonious models by shrinking all coefficients toward zero, nevertheless, these methods fail to reveal the correlated structure among the active variables. Moreover, in the

presence of unknown groups of variables, it would be preferable to have a procedure that selectively shrinks the coefficients of the same group toward each other rather than toward zero.

In this work, we propose a unified penalized SVM framework which uses a Cluster Correlation-Network (CCN) penalty in combination with Lasso, SCAD or MCP penalty. We called it the Cluster Correlation-Network SVM (CCNSVM). The CCNSVM framework is a two-in-one step procedure which uses CCN penalty in order to cluster predictors in groups that are relevant to the classification variable and perform variable selection simultaneously. For the same objective in regression setting, Witten *et al.* (2014) considered a relatively similar cluster penalty with ℓ_1 norm in their Cluster Elastic Net (CEN) approach. Although Witten's CEN penalty takes into account the correlation between the predictors, in certain situations correlation may not be enough to capture more complex group structure, especially, in the presence of a relationship networking between the predictors. Capturing the predictors' relationship networking is a desirable property when analyzing, for instance, gene expression and/or DNA methylation data. In contrast, our CCN penalty is a function of the well-known Topological Overlap Matrix (TOM) whose general term measures the strength of connectivity between two predictors, Zhang et Horvath (2005). Our motivation comes from the great success of the TOM matrix for detecting gene interaction networks or social interaction networks. In addition, we propose an efficient and simple implementation algorithm for obtaining the penalized SVM coefficients and clusters of the predictors. The algorithm alternates between two steps until convergence : in the first step, a search for clustering the predictors into K groups is obtained, then the second step consists of solving the optimization problem, with fixed groups from step one, using Majorization-Minimization principle together with a coordinate descent algorithm as in Yang et Zou (2013) for EnetSVM and Mkhadri *et al.* (2017) in penalized quantile regression.

In Section 4.2, we review the SVM approach and its penalized versions proposed recently in high-dimension binary classification settings. In Section 4.3, we present our CCNSVM procedure, which is based on the combination of CCN penalty and Lasso, SCAD or MCP penalty. Some properties of CCN penalty are presented and an efficient implementation algorithm for obtaining the penalized SVM coefficients and predictors' clusters is detailed. Simulation studies to evaluate the performance of the proposed methodology are considered in Section 4.4. The use of our framework is illustrated through analysis of bisulfite sequencing DNA methylation data in Section 4.5. We end with a brief discussion in Section 2.6.

2.2 Penalized SVM

In this section we give a brief review of the penalized SVM framework for classification of high-dimensional data. A detailed discussion of SVM and its Kernel version can be found in Hastie *et al.* (2009). Assume we observe a training data of n pairs, $\mathcal{T} = \{(y_1, \boldsymbol{x}_1), \ldots, (y_n, \boldsymbol{x}_n)\}$, where $\boldsymbol{x}_i \in I\!R^p$ and $y_i \in \{-1, 1\}, i = 1, ..., n$, denotes class labels.

2.2.1 SVM method

The SVM method Cortes et Vapnik (1995) seeks for a linear boundary hyperplane to separate the training data $\mathcal{T}$ into classes 1 and -1. This hyperplane is the solution to the following minimization problem

$$\begin{aligned} &\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{\beta}||_2^2 + \mu \sum_{i=1}^{n} \xi_i \\ &\text{subject to} \quad \xi_i \geq 0, \quad y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq 1 - \xi_i, \quad i = 1, ..., n, \end{aligned} \tag{2.1}$$

where $\xi_1, ..., \xi_n$ are slack variables, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\top$, $\top$ stands for the transpose and μ is a positive constant. The optimisation problem (2.1) is known as the soft margin SVM. The constant μ is a tuning parameter which depends on the data at hand and its optimal value searches to control for the balance between minimizing the coefficients of the hyperplane and correct classification of the training data set. Optimal μ can be obtained using cross validation. Problem (2.1) is quadratic with linear inequality constraints, hence it is a convex optimization problem which can be solved by quadratic programming using Lagrange multipliers Gunn *et al.* (1998) .

2.2.2 SVM as a penalized approach

Hastie *et al.* (2009) showed that the problem in (2.1) is equivalent to the following Ridge penalized problem

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} h(y_i, (\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})) + \frac{\lambda_2}{2}||\boldsymbol{\beta}||_2^2, \tag{2.2}$$

where λ_2 is a positive tuning parameter and $h(a, b)$ is the check function defined as

$$h(a, b) = (1 - ab)_+,$$

with

$$(1-t)_+ = \begin{cases} 0, & t > 1 \\ 1, & t \leq 1. \end{cases}$$

Although the Ridge penalty shrinks the coefficient estimates in order to reduce their variances, it does not provide sparse solutions for variable selection ; such a behaviour is not suitable in high-dimensional settings. Consequently, several extensions of penalized SVM, for simultaneously variable selection and estimation of the coefficients, are developed hereafter with most known penalties.

In its general form, the penalized SVM problem is given by

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n} h(y_i, (\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})) + p_{\lambda_1,\lambda_2}(\boldsymbol{\beta}), \tag{2.3}$$

where $p_{\lambda_1,\lambda_2}(\boldsymbol{\beta})$ stands for a general penalty term that may depend on positive regularization parameters λ_1 and/or λ_2.

The penalty term $p_{\lambda_1,\lambda_2}(\boldsymbol{\beta})$ can take different forms. For instance, Bradley et Mangasarian (1998) adapted the L1-norm penalty to the SVM method leading to the ℓ_1-SVM method which solves problem (3.6) with

$$p_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) = \lambda_1 ||\boldsymbol{\beta}||_1.$$

However, this approach doesn't take into account the correlation between the predictors and tends to select only one predictor from a group in the presence of highly correlated predictors (i.e., groups of predictors). To overcome such a problem, Wang *et al.* (2006) proposed the Hybrid Huberized Support Vector Machine (HHSVM). The authors adapted the Elastic Net penalty to SVM (HHSVM+EN) by solving problem (2.2) with

$$p_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) = \lambda_1 ||\boldsymbol{\beta}||_1 + \frac{\lambda_2}{2} ||\boldsymbol{\beta}||_2^2.$$

The authors showed that HHSVM+EN tends to select significant variables, and it takes into account the correlation between predictors, but it shrinks all coefficients towards zero.

Lasso penalty lacks the selection consistency property because it tends to overly shrink the relevant variables. To overcome the over-shrinkage problem and gain selection consistency, Becker *et al.* (2011) have suggested the SCAD-SVM method which solves the optimization problem (3.6) with

$$p_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) = \sum_{j=1}^{p} p_{\lambda_1}^{SCAD}(\beta_j),$$

where $p_{\lambda_1}^{SCAD}(.)$ is defined by

$$p_{\lambda_1}^{SCAD}(\beta_j) = \lambda_1|\beta_j|\mathbb{1}_{|\beta_j|\leq\lambda_1} - \frac{|\beta_j|^2 - 2a\lambda_1|\beta_j| + \lambda_1^2}{2(a-1)}\mathbb{1}_{\lambda_1<|\beta_j|\leq a\lambda_1} + \frac{(a+1)\lambda_1^2}{2}\mathbb{1}_{|\beta_j|>a\lambda_1},$$

with $a > 2$ and $\mathbb{1}_A(x) = 1$ if $x \in A$ and 0 otherwise.

In order to keep the advantages of the SCAD penalty, and at the same time, avoid too restrictive sparsity limitations for non-sparse data, Becker *et al.* (2011) suggested also SVM with SCAD + EN penalty analogously to Elastic Net (i.e. SCAD + L2-norm).

Although SVM with SCAD elastic net penalty is implemented in the `R` package software "PenalizedSVM", it is implemented for a single λ_1 value at a time, and thus it does not benefit from the use of the warm-start and the active-set tricks that can be used when calculating path solutions for a grid of λ_1.

All these penalized approaches select correlated relevant variables and produce parsimonious models by shrinking all coefficients toward zero. However, they fail to reveal the correlated structure among the active variables. Moreover, in the presence of unknown groups of variables, it would be preferable to have a procedure that shrinks the coefficients of the same group toward each other rather than toward zero.

In the next section, we propose a unified penalized SVM framework which uses a Cluster Correlation-Network (CCN) penalty in order to cluster the predictors in groups that are relevant to the response variable and perform group variable selection simultaneously. We have called the latter approach the Cluster Correlation-Network SVM (CCNSVM).

2.3 The CCNSVM Method

2.3.1 Penalized SVM with CCN penalty

In the sequel, we assume that the matrix of predictors $\boldsymbol{X}$ is standardized. That is, we assume

$$\frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0, \quad \frac{1}{n}\sum_{i=1}^{n} x_{ij}^2 = 1, \quad j = 1, ..., p.$$

We also assume that $p >> n$ and the columns of $\boldsymbol{X}$ consist of K distinct and *unknown* groups

$C_1, ..., C_K$, with $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, 2, \ldots, p\}$.

The CCNSVM framework solves the following optimization problem for $(C_1, ..., C_K)$ and $(\beta_0, \boldsymbol{\beta})$

$$\min_{\{C_1,...,C_K,\beta_0,\boldsymbol{\beta}\}} \frac{1}{n}\sum_{i=1}^{n} h(y_i, (\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})) + p_{\lambda_1}(\boldsymbol{\beta}) + \frac{\lambda_2}{2}\sum_{k=1}^{K}\frac{1}{|C_k|}\sum_{j,\ell \in C_k} ||\boldsymbol{L}_j\beta_j - \boldsymbol{L}_\ell\beta_\ell||^2, \qquad (2.4)$$

where $p_{\lambda_1}(\boldsymbol{\beta})$ could be either Lasso, SCAD or MCP penalty, with the MCP penalty is introduced by Zhang (2010), and is given explicitly in equation (2.17) of Appendix **A**. $\boldsymbol{L}_j, j = 1, \ldots, p$ is the jth column of the $p \times p$ matrix $\boldsymbol{L}$, with $\boldsymbol{L}$ derived from Cholesky decomposition of the $p \times p$ Topological Overlap Matrix (TOM), $\boldsymbol{\Omega}$, such that $\boldsymbol{\Omega} = [\Omega_{j\ell}] = \boldsymbol{L}^\top \boldsymbol{L}$. The entries $\Omega_{j\ell}$ of the TOM matrix are defined as

$$\Omega_{j\ell} = \frac{d_{j\ell} + |\rho_{j\ell}|^b}{min\{c_j, c_\ell\} + 1 - |\rho_{j\ell}|^b},$$

with $\rho_{j\ell} = \boldsymbol{x}_j^\top \boldsymbol{x}_\ell / n$ is the empirical Pearson correlation between the pair of predictors (j, ℓ), $d_{j\ell} = \sum_{u \neq j} |\rho_{ju}|^b |\rho_{u\ell}|^b$ and $c_j = \sum_{u \neq j} |\rho_{ju}|^b$, with $1 \leq u \leq p$.

The coefficient $|\rho_{j\ell}|^b$ is called the adjacency measure between predictors (j, ℓ) and it measures the strength of connectivity between the two predictors. The positive coefficient b is a soft-threshold power whose value is fixed based on a biologically motivated criterion (referred to as scale-free topology criterion). The optimal choice of b is discussed extensively in Zhang et Horvath (2005) and can be obtained using the R package software *WGCNA*. The coefficient $d_{j\ell}$ reflects (non-standardized) similarity between j and ℓ predictors (nodes) in terms of the commonality of the nodes they connect to. For instance, in the case of hard thresholding (i.e. $|\rho_{u\ell}|^b$ is 0 or 1), $d_{j\ell}$ counts the number of nodes to which both j and ℓ are connected. On the other hand, as demonstrated in Zhang et Horvath (2005), $0 \leq \Omega_{j\ell} \leq 1$, and so, $\Omega_{j\ell}$ provides a (standardized) similarity measure which reflects the relative interconnectedness between predictors (j, ℓ). The TOM matrix has been found useful in biological networks (Ravasz *et al.* (2002) ; Ye et Godzik (2004). Moreover, for clustering of gene expression profiles, Zhang et Horvath (2005) showed that the TOM-based clustering leads to more distinct gene clusters than the current standard correlation-based clustering.

The penalty term $p_{\lambda_1}(\boldsymbol{\beta})$ in (2.4) encourages sparsity in the coefficient estimates when λ_1 is large. However, the impact of the Cluster Correlation-Network penalty term in (2.4) is more subtle and is discussed in more detail in the next Section.

This work is motivated by the work of Witten *et al.* (2014), who proposed the cluster elastic net (CEN) approach that simultaneously estimates clusters of covariates and fuses the effects of covariates within the same cluster in linear multiple regression. However, their cluster penalty is similar to the form of CCN penalty with $\boldsymbol{\Omega} = \mathbf{X}^\top \mathbf{X}/n$, which is the predictors' empirical correlation matrix.

2.3.2 Properties of the Cluster Correlation-Network penalty

Assume for a moment that the clusters $C_1, ..., C_K$ are fixed/known. To illustrate some properties of the CCN penalty and its behaviour in (2.4), one can verify first that the Cluster Correlation-Network part of the penalty in (2.4) can be written as

$$\frac{\lambda_2}{2}\sum_{k=1}^{K}\frac{1}{|C_k|}\sum_{j,\ell\in C_k}||\boldsymbol{L}_j\beta_j - \boldsymbol{L}_\ell\beta_\ell||^2 = \frac{\lambda_2}{2}\sum_{k=1}^{K}\frac{1}{|C_k|}\sum_{j,\ell\in C_k}\big[(1-\Omega_{j\ell})(\beta_j^2+\beta_\ell^2)+\Omega_{j\ell}(\beta_j-\beta_\ell)^2\big].$$

Consequently, for relatively large values of λ_2, solving the CCNSVM problem encourages the coefficients of predictors belonging to the same class to take approximately on similar values, if the corresponding predictors (j, ℓ) are highly connected (i.e. with large values of $\Omega_{j\ell}$). More precisely, if $\Omega_{j\ell}$ is close to 1, then $(\beta_j - \beta_\ell)^2$ will dominate in (2.4), and CCN will shrink β_j and β_ℓ towards each other, $\beta_j \simeq \beta_\ell$. On the other hand, if $\Omega_{j\ell}$ is close to zero, $(\beta_j^2 + \beta_\ell^2)$ will dominate, and CCN will shrink β_j and β_ℓ towards zero, which leads to a ridge-type penalty on a subset of the predictors.

Of note that in some situations, achieving group structure based on network clustering is not a desirable property for the predictors. In such situations, we suggest to change the similarity matrix in the CCN penalty in order to capture the desired group structure of the predictors. On the other hand, if capturing the signed correlation between predictors is of interest, for instance, then one can set $\boldsymbol{\Omega} = \mathbf{X}^\top \mathbf{X}/n$. This leads to the following straightforward result

$$\frac{1}{2}\sum_{k=1}^{K}\frac{1}{|C_k|}\sum_{j,\ell\in C_k}||\boldsymbol{L}_j\beta_j - \boldsymbol{L}_\ell\beta_\ell||^2 = \frac{1}{2}\sum_{k=1}^{K}\frac{1}{|C_k|}\sum_{j,\in C_k}||\boldsymbol{X}_j\beta_j - \boldsymbol{X}_\ell\beta_\ell||^2. \tag{2.5}$$

This is exactly the Cluster Elastic Net penalty proposed by Witten *et al.* (2014) for penalized least-squares regression. Thus, our framework generalizes the work of Witten *et al.* (2014) to penalized SVM classification/regression method. The work of Witten *et al.* (2014) coupled L1-norm with the Cluster Elastic Net penalty given in (2.5) for least squares regression. In contrast, here we allow the sparse-term of the CCNSVM framework to be the L1-norm, SCAD or MCP penalty.

Since we do not know prior information concerning the clustering of the predictors, we are making the assumption that better capturing the class structure in predictors may allow us to gain more accuracy when estimating the coefficients $\boldsymbol{\beta}$, which in its turn may lead to improve class prediction. The CCNSVM procedure estimates clusters of predictors and coefficients simultaneously. Changes in clustering classes are discrete changes and as a result the objective function (2.4) is discontinuous. So, as in Witten *et al.* (2014), it is difficult to derive an efficient algorithm to obtain the optimal estimates of coefficients and classes. Consequently, the penalized CCNSVM framework solves problem (2.4) by iterating between two steps until convergence : in step 1, a search for clustering the predictors into K groups is first obtained, then step 2 consists of solving (2.4) with fixed groups from step 1, using Majorization-Minimization principle together with a coordinate descent algorithm. More details about this two-step iterative algorithm is given next in Section 2.3.3.

2.3.3 The CCNSVM Algorithm

We are facing two main obstacles when solving (2.4) :

The first issue is that the problem (2.4) is not convex and searching for global optimum requires considering all $O(K^p)$ possible partitions of p predictors in K classes. Then for each fixed partition, the problem (2.4) needs to be solved. Since we are dealing with large p, this approach is not feasible. To remedy this issue one can solve (2.4) using a block-relaxation algorithm which iterates between two steps : in the first step one can fix the classes and solve (2.4) with respect to $\boldsymbol{\beta}$ and in the second step one can fix the $\boldsymbol{\beta}$ and solve (2.4) with respect to $C_1, ..., C_K$. Solving (2.4) with respect to $C_1, ..., C_K$ is equivalent to solving

$$\min_{C_1,...,C_K} \frac{1}{2} \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{j,\ell \in C_k} ||\boldsymbol{L}_j \beta_j - \boldsymbol{L}_\ell \beta_\ell||^2,$$

which in turn is equivalent to solve the following problem

$$\min_{C_1,...,C_K} \sum_{k=1}^{K} \sum_{j \in C_k} ||\boldsymbol{L}_j \beta_j - \frac{1}{|C_k|} \sum_{\ell \in C_k} \boldsymbol{L}_\ell \beta_\ell||^2. \tag{2.6}$$

This is exactly the formulation of the K-means optimization problem (Hartigan et Wong (1979)) for clustering columns of the matrix $[\boldsymbol{L}_1\beta_1; ...; \boldsymbol{L}_p\beta_p]$.

The second issue in solving problem (2.4) is due to the non-differentiability of the SVM hinge loss function, $\mathcal{L}(t) = [1-t]_+$, at $t = 1$, which leads to computational difficulties in solving (2.4) with

respect to $\boldsymbol{\beta}$. To overcome this issue, we suggest to approximate the SVM hinge loss by the Hybrid Huberized SVM loss function (HHSVM) proposed by Yang et Zou (2013)

$$\phi_\delta(t) = \begin{cases} 0, & t > 1 \\ \frac{(1-t)^2}{2\delta}, & 1-\delta < t \leq 1 \\ 1 - t - \frac{\delta}{2}, & t \leq 1-\delta, \end{cases}$$

where $\delta > 0$ is a positive constant.

The HHSVM loss function $\phi_\delta(t)$ is differentiable everywhere and its first derivative is continuous. Moreover, for small values of δ, the two loss functions $\phi_\delta(t)$ and $\mathcal{L}(t)$ become very similar. The HHSVM loss function is used in Yang et Zou (2013) to solve penalized SVM with Elastic Net penalty and groupe Lasso, and the authors showed its computational gain efficiency compared to using the standard SVM hinge loss function.

In summary, the two-step strategy for solving (2.4) is given as follows. For fixed $\boldsymbol{\beta}$, solve K-means clustering problem (2.6) for $C_1, ..., C_K$, then fix the classes as the K-means solution and solve the following problem with respect to $\boldsymbol{\beta}$

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} \phi_\delta(y_i(\beta_0 + \mathbf{x}_i^T \boldsymbol{\beta})) + p_{\lambda_1}(\boldsymbol{\beta}) + \sum_{k=1}^{K} \sum_{j \in C_k} ||\boldsymbol{L}_j \beta_j - \frac{1}{|C_k|} \sum_{\ell \in C_k} \boldsymbol{L}_\ell \beta_\ell||^2. \tag{2.7}$$

Of note that K-means problem leads to a local optimum solution. Thus, we are warned that our two-step strategy may not lead to a global minimum of the whole problem. However, to solve (2.7) for $\boldsymbol{\beta}$, we rely on the MM trick and a coordinate descent algorithm that guarantee the descent property of the objective function, at each iteration. More details about the algorithm are given next.

2.3.4 Details of CCNSVM Algorithm

This section gives details about the CCNSVM algorithm.

For fixed classes $C_1, \ldots, , C_k$, solving (2.7) with respect to $\boldsymbol{\beta}$ can be done efficiently using a coordinate descent algorithm as in Yang et Zou (2013). To do so, define the current margin $r_i = y_i(\tilde{\beta}_0 + \mathbf{x}_i^\top \tilde{\boldsymbol{\beta}})$, with $\tilde{\boldsymbol{\beta}}$ is the update of $\boldsymbol{\beta}$ from the last iteration. For the jth predictor, $j = 1, \ldots, p$, assume that

$j \in C_k$ for a fixed class k, and define

$$\mathcal{L}(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) = \frac{1}{n}\sum_{i=1}^{n}\phi_\delta(r_i + y_i x_{ij}(\beta_j - \tilde{\beta}_j)) + p_{\lambda_1}(\beta_j) + \frac{\lambda_2}{2|C_k|}\sum_{j,\ell \in C_k}||\boldsymbol{L}_j\beta_j - \boldsymbol{L}_\ell\tilde{\beta}_\ell||^2. \tag{2.8}$$

One can cycle through the coefficients $j = 1, \ldots, p$, updating only one parameter β_j at each stage of a cycle by minimizing (2.8) with respect to β_j. However, the major difficulty in solving the above coordinate descent procedure is that the univariate minimization problem in (2.8) does not have a closed form solution.

To circumvent such a problem, we take advantage of the Lipchitz property of the first derivative of the HHSVM loss function and we use the mean-value Theorem to build the following easy-to-minimize surrogate loss function which approximates the objective function $\mathcal{L}(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$ in (2.8), (see also Yang and Zou, 2013)

$$Q(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) = \frac{\sum_{i=1}^{n}\phi_\delta(r_i)}{n} + \frac{\sum_{i=1}^{n}\phi'_\delta(r_i)y_i x_{ij}}{n}(\beta_j - \tilde{\beta}_j) + \frac{1}{\delta}(\beta_j - \tilde{\beta}_j)^2 + p_{\lambda_1,\lambda_2}(\beta_j), \tag{2.9}$$

where $\phi'_\delta(t)$ is the first derivative of $\phi_\delta(t)$. Thus, one can update β_j as follows

$$\tilde{\beta}_j^{\text{new}} \leftarrow arg\min_{\beta_j} Q(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}). \tag{2.10}$$

The minimizer of (2.10) with respect to β_j has a soft-threshold explicit form, which allows for a substantial gain of computational time in solving our CCNSVM problem. The soft-threshold explicit form depends on the sparse penalty, $p_{\lambda_1}(.)$, used in solving (2.10). The next Proposition gives the three explicit forms of the update $\tilde{\beta}_j^{\text{new}}$ for CCNSVM when $p_{\lambda_1}(.)$ is the Lasso, Scad or MCP penalty.

Proposition 2.1 *Let $Q(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$ be the surrogate loss function defined in (2.9). Let $j \in C_k$, $j = 1, \ldots, p$ for some $k \in \{1, \ldots, K\}$. Let $p_{\lambda_1}(.)$ be the L_1-norm, SCAD or MCP penalty. The closed form solution of $\tilde{\beta}_j^{\text{new}}$, the minimizer of (2.10), is given as follows*

$$\hat{\beta}_j^{Lasso} = \frac{S(Z_j, \lambda_1)}{\omega}, \tag{2.11}$$

$$\begin{aligned}\hat{\beta}_j^{Scad} &= \frac{1}{\omega}S\left(Z_j, \lambda_1\right)\mathbb{1}_{\{|Z_j| \le (1+\omega)\lambda_1\}} + \frac{1}{\omega}Z_j\mathbb{1}_{\{|Z_j| > a\omega\lambda_1\}} \\ &+ \frac{a-1}{\omega(a-1)-1}S\left(Z_j, \frac{a\lambda_1}{(a-1)}\right)\mathbb{1}_{\{(1+\omega)\lambda_1 < |Z_j| \le a\lambda_1\omega\}},\end{aligned} \tag{2.12}$$

$$\hat{\beta}_j^{Mcp} = \frac{\gamma}{\gamma\omega - 1} S(Z_j, \lambda_1) \mathbb{1}_{\{|Z_j| < \gamma\lambda_1\omega\}} + \frac{1}{\omega} Z_j \mathbb{1}_{\{|Z_j| \geq \gamma\lambda_1\omega\}}, \tag{2.13}$$

where $S(.,.)$ *is the soft-thresholding function defined by*

$$S(a, b) = (|a| - b)_+ \; sign(a),$$

$\omega = \{\frac{2}{\delta} + \frac{\lambda_2(|C_k|-1)}{|C_k|}\}$, *and* $Z_j = (\frac{2}{\delta}\tilde{\beta}_j + \frac{\lambda_2 \sum_{\ell \in C_k, \ell \neq j} \Omega_{j\ell}\tilde{\beta}_\ell}{|C_k|} - \frac{\sum_{i=1}^n \phi'_\delta(r_i) y_i x_{ij}}{n})$.

The proof of Proposition 2.1 is postponed to Appendix **A**.

To sum up, we propose a two-step algorithm for solving CCNSVM which uses a K-means algorithm in a first step to cluster the p predictors in K classes, and in a second step it implements a coordinate descent algorithm and MM tricks to efficiently update the coefficient estimates. The details of the algorithm are given next.

Algorithm 2 The CCNSVM algorithm for Lasso-, Scad- and Mcp-CCN penalties

1. Calculate the TOM matrix $\boldsymbol{\Omega}$ and compute its Cholesky decomposition $\boldsymbol{L}$;
2. Initialize $\tilde{\beta}_0$ and $\tilde{\boldsymbol{\beta}}$ as the solution to the SVM Elastic Net approach

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n} h(y_i,(\beta_0+\boldsymbol{x}_i^\top\boldsymbol{\beta}))+\lambda_1||\boldsymbol{\beta}||_1+\frac{\lambda_2}{2}||\boldsymbol{\beta}||_2^2;$$

3. Iterate the following updates until convergence :
   (a) Solve the following K-means algorithm for $C_1,\ldots,C_K$

$$\min_{C_1,\ldots,C_K}\sum_{k=1}^{K}\sum_{j\in C_k}||\boldsymbol{L}_j\tilde{\beta}_j-\frac{1}{|C_k|}\sum_{\ell\in C_k}\boldsymbol{L}_\ell\tilde{\beta}_\ell||^2;$$

   (b) For fixed groups $C_1,\ldots,C_K$
   — For $j=1,\ldots,p$, update β_j
   ∗ compute $r_i=y_i(\tilde{\beta}_0+\mathbf{x}_{i.}^\top\tilde{\boldsymbol{\beta}})$;
   ∗ set

$$\tilde{\beta}_j^{\text{new}}\longleftarrow\hat{\beta}_j^*,$$

   with $\hat{\beta}_j^*$ is given by (2.11), (2.12) or (2.13), corresponding to the three sparse penalty terms L_1-norm, $p_{\lambda_1}^{Scad}(.)$ and $p_{\lambda_1}^{Mcp}(.)$ respectively;
   — Update β_0
   ∗ compute $r_i=y_i(\tilde{\beta}_0+\mathbf{x}_{i.}^\top\tilde{\boldsymbol{\beta}})$,
   ∗ set

$$\tilde{\beta}_0^{\text{new}}\longleftarrow\tilde{\beta}_0-\frac{\delta}{2}\frac{\sum_{i=1}^{n}\phi_\delta'(r_i)y_i}{n}.$$

It is important to note that

$$\mathcal{L}(\beta_j|\tilde{\beta}_0,\tilde{\boldsymbol{\beta}})=Q(\beta_j|\tilde{\beta}_0,\tilde{\boldsymbol{\beta}}),\qquad\text{if}\quad\beta_j=\tilde{\beta}_j$$

$$\mathcal{L}(\beta_j|\tilde{\beta}_0,\tilde{\boldsymbol{\beta}})\leq Q(\beta_j|\tilde{\beta}_0,\tilde{\boldsymbol{\beta}}),\qquad\text{if}\quad\beta_j\neq\tilde{\beta}_j.$$

Therefore, in step 3(b) of Algorithm 2, the update of $\tilde{\beta}_j^{\text{new}}$ is based on minimizing the quadratic form $Q(\beta_j|\tilde{\beta}_0,\tilde{\boldsymbol{\beta}})$ which is a majorizing function of $\mathcal{L}(\beta_j|\tilde{\beta}_0,\tilde{\boldsymbol{\beta}})$. After updating all parameters, the theory underlying MM algorithms ensures that the CCNSVM algorithm, which consists of alternately executing the majorizing step and the minimization steps, retains the descent property of the objective function in (2.8).

2.3.5 Implementation and choice of tuning parameters and the number of classes K

Implementation : Our approach is implemented in an R package software "CCNSVM" available publicly in github : `github\OulkachaKarim\CCNSVM`.

The CCNSVM model is solved by using a fine grid of λ_1 and λ_2. For each fixed λ_2 we proceeded by choosing $\lambda_{\max}$ which is the smallest λ_1 to let all predictors, $\beta_j,\ (1 \leqslant j \leqslant p)$ to be zero, except the intercept. To obtain $\lambda_{\max}$, we first obtained estimates, $\hat{y}$, for the null model with only the intercept :

$$\hat{y} = arg\min_{y} \frac{1}{n}\sum_{i=1}^{n} \phi_\delta(y_i y).$$

According to KKT conditions, we can obtain the following formula

$$\lambda_{\max} = \frac{1}{n}\max_{j} |\sum_{i=1}^{n} \phi_\delta^{'}(\hat{y}) y_i x_{ij}|.$$

Let $\lambda_{\min} = \eta\lambda_{\max}$, where $0 < \eta < 1$ is a small number. We generated a sequence of λs $\{\lambda_1^{[l]}\}_{l=1}^{100}$ by placing 98 evenly spaced points in log-scale, $\{\lambda^{[l]}\}_{l=2}^{99}$, between $\lambda_{\max}$ and $\lambda_{\min}$ and let $\lambda^{[1]} = \lambda_{\max}$ and $\lambda^{[100]} = \lambda_{\min}$.

To speed up computations in our implementation, we used the warm-start and the active set tricks Friedman *et al.* (2010), Yang et Zou (2013) in solving the solution paths in Step 3(b) of Algorithm 2. For the warm-start trick, the solution of $\widehat{\boldsymbol{\beta}}$ at $\lambda_1^{[l-1]}$ is taken as the initial value for solving the solution of $\widehat{\boldsymbol{\beta}}$ at $\lambda_1^{[l]}$. For the active-set cycling idea, we only apply coordinate descent until convergence on the active-set, defined as the set contains variables with nonzero coefficients at the current iteration.

The convergence criterion used in Algorithm 2 suggests iteration between Steps 3(a) and 3(b) of Algorithm 2 until the relative change in the estimated coefficients, $||\widehat{\boldsymbol{\beta}}^{old} - \widehat{\boldsymbol{\beta}}^{new}||^2/||\widehat{\boldsymbol{\beta}}^{new}||^2 < \epsilon = 10^{-5}$.

Choice of λ_1 and λ_2 : To select the values of the tuning parameters λ_1 and λ_2, CCNSVM implements two-dimensional five-fold cross-validation to find the best pair of (λ_1, λ_2) that gives minimal misclassification error.

Choice of K : In the case where no prior knowledge is available for the value of K, the selection

of the number of groups can be also determined using cross-validation. In such a case, in order to accelerate the computations in the CCNSVM framework, we suggest the following work-flow to select K : i) use formal statistical tests provided by some unsupervised clustering methods to choose the optimal K_{opt} to be retained. ii) conduct a five-fold-CV for a grid of values of K around K_{opt} obtained in step i) and retain the value that incurs minimal misclassification error. Of note that in step i), one can use for instance the Gap test statistic to determine K_{opt} (Tibshirani *et al.* (2001). Several unsupervised-clustering test-statistics which deal with this issue, including the Gap statistic, are implemented in two `R` software packages : `NbClust` (CHARRAD *et al.* (2014)) and `cluster` (Rosseeuw et Van Zomeren (1990).

2.4 Simulation study

We carried out two simulation scenarios in order to evaluate the performance of our proposed methodology. In the two simulation scenarios we compared our framework with existing competitors.

2.4.1 Simulation study designs

Scenario 1 In this scenario we generated data with $n = 200$ observations and $p = 1000$ predictors from the following model :

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where the rows of the matrix $\boldsymbol{X}$ are generated from a multivariate normal distribution $\mathbf{N}(\mathbf{0_p}, \boldsymbol{\Sigma})$, such that $\boldsymbol{\Sigma} \in \mathbb{M}_{p\times p}$ is a block diagonal matrix defined by

$$\Sigma_{j\ell} = \begin{cases} 1 & \text{if} \quad j = \ell \\ \rho & \text{if} \quad j \leq 50, \ell \leq 50 \quad \text{and } j \neq \ell \\ \rho & \text{if} \quad 51 \leq j \leq 100, 51 \leq l \leq 100 \text{ and } j \neq \ell \\ 0 & \text{otherwise,} \end{cases}$$

with ρ denotes the correlation parameter between predictors. The elements of $\boldsymbol{\epsilon}$, ϵ_i's, for $i = 1, \ldots, n$, are simulated independently following a normal distribution with mean zero and variance $\sigma^2 = 2.5^2$. The vector of coefficients, $\boldsymbol{\beta}$, takes the form

$$\begin{cases} \beta_j \sim Unif[0.9, 1.1], & \text{if} \quad 1 \leq j \leq 25 \\ \beta_j \sim Unif[-1.1, -0.9], & \text{if} \quad 51 \leq j \leq 75 \\ \beta_j = 0 & \text{otherwise.} \end{cases}$$

More precisely, we have two sets of 50 correlated predictors, 25 of them are associated with the response variable and the other 25 are noisy variables. The remaining 900 predictors are simulated independently of the response variable and the first 100 features. Note that for clusters 1 and 2, not all the β's are similar within each cluster ; Scenario 1 tends to mimic gene pathways, where genes within the same pathway have correlated expression values, however only a portion of genes in the pathway have expression that is associated with a response of interest.

Finally, the binary response Y is generated according to the following relationship with Z :

$$I\!P(Y = -1) = \frac{1}{1 + \exp(-Z)}, \quad \text{and} \quad I\!P(Y = 1) = \frac{1}{1 + \exp(Z)}. \tag{2.14}$$

Scenario 2 This scenario aimed to test the utility of using the TOM matrix in the CCN penalty to better capture gene network clustering and to provide accurate prediction of the binary response (e.g. disease status). That is, we generated the data based on the WGCNA model, which tends to mimic gene-expression real-data situations and the network connectivity for a disease binary response. To do so, we simulated data as described in the tutorial of Langfelder et Horvath (2014). Specifically, we used the WGCNA R function, *simulateModule*, to simulate the groups of predictors. This function generates gene-expression-like data sets that exhibit a modular structure (i.e. group structure) similar to that observed in real gene expression data.

We fixed the number of individuals to be $n = 200$ with 100 diseased subjects and 100 controls. We set the number of genes to be $p = 1000$, and the number of the groups $K = 6$, with $|C_1| = 25$, $|C_2| = 25$, $|C_3| = 50$, $|C_4| = 25$, $|C_5| = 25$ and $|C_6| = 850$, where $|C|$ is the cardinal of the set C.

The binary response is only associated with the first two groups of predictors C_1 and C_2. The groups C_3 and C_4 are not associated with the binary response but are correlated to each other. The C_5 is generated independently of the disease and not correlated to groups C_1, C_2, C_3 and C_4, but its predictors are correlated. The noisy group, C_6, consists of predictors that are generated independently to each other and independent with the disease/binary response variable and all the other groups. The WGCNA procedure proceeds in two steps to generate such a structure of the data. In the first step, WGCNA simulates a continuous response variable Z and five variables $c_1, c_2, \ldots, c_5$, with $corr(Z, c_1) = 0.8$, $corr(Z, c_2) = -0.8$ and $corr(c_3, c_4) = 0.6$, as follows

Step 1 :

- generate $c_1 \sim N(0,1)$,
- set $Z = 0.8\ c_1 + \sqrt{(1-0.8^2)}\ e_1$, with $e_1 \sim N(0,1)$,
- simulate $c_2 = -0.8\ Z + \sqrt{(1-0.8^2)}\ e_2$, with $e_2 \sim N(0,1)$,
- generate $c_3 \sim N(0,1)$,
- set $c_4 = 0.6\ c_3 + \sqrt{(1-0.6^2)}\ e_3$, with $e_3 \sim N(0,1)$,
- generate $c_5 \sim N(0,1)$.

In the second step, WGCNA uses the five variables c_k, $k = 1, \ldots, 5$, to construct the five predictors' sub-matrices, $\boldsymbol{X}_k$, $k = 1, \ldots, 5$, as follows

Step 2 :

- For $k = 1, \ldots, 5$, for $j = 1, \ldots, |C_k|$, generate $x_j^{(k)} = c_k + w_j$,

with $w_j \sim N(0, s_j)$ and $s_j = \sqrt{\text{var}(c_k)(\frac{1-\alpha_j^2}{\alpha_j^2})}$, and $\alpha_j = 1 - \frac{j}{|C_k|}0.7, \quad j = 1, ..., |C_k|$.

All noisy predictors of group C_6 are generated as $x_j^{(6)} \sim N(0,1)$, for $j = 1, \ldots, |C_6|$.

The binary response Y is then generated from Z according to the relationship given in (2.14). Note that by construction one has $corr(x_j^{(k)}, c_k) = 1 - j/|C_k|(1-0.7)$, for $k = 1, \ldots, 5$. This technique produces groups consisting of predictors distributed symmetrically around c_k. In this sense, the simulated groups are spherical clusters whose centers coincide (on average) with c_k. Figure 2.1 summarizes the six group structures using the hierarchal clustering approach implemented in WGCNA R package. In this figure, each group is represented by a color : green (C_1), brown (C_2), yellow (C_3), turquoise (C_4), blue (C_5) and gray (C_6).

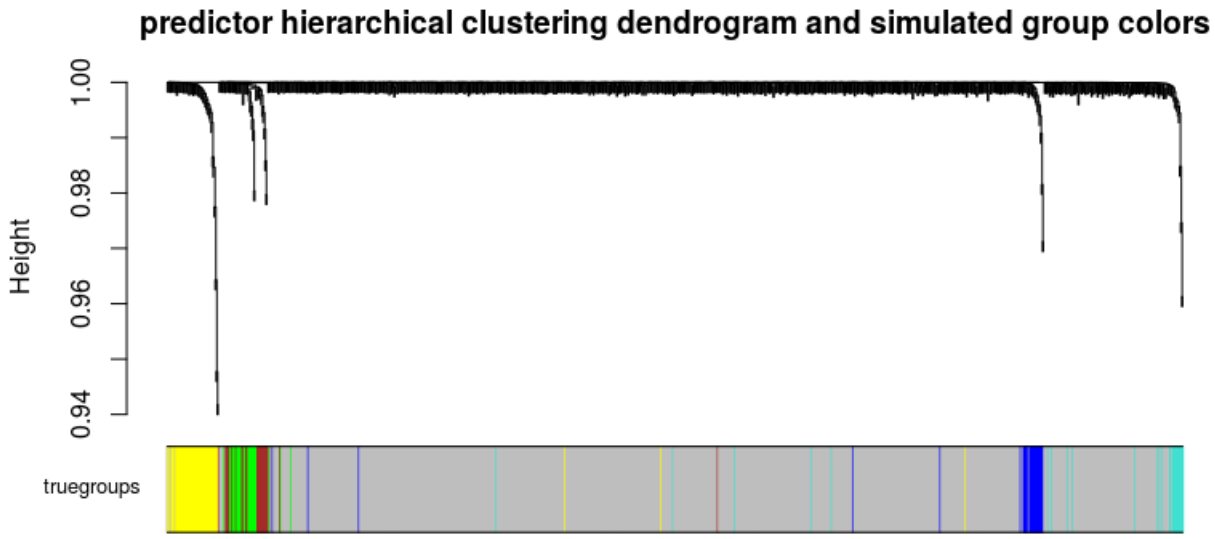

FIGURE 2.1 The six group structures of WGCNA simulated data used in Scenario 2 of our simulation set-up are shown. The dendrogram is obtained using the hierarchal clustering approach implemented in WGCNA R package. Predictors belonging to the same group are represented with a same color : green for predictors of C_1, brown for C_2, yellow for C_3, turquoise for C_4, blue for C_5, and gray for C_6.

2.4.2 Simulation results

In both scenarios, we compared our approach CCNSVM with CCN+L1, CCN+SCAD and CCN+MCP penalties and the penalized SVM method implemented in the `R` package "gcdnet", Yang et Zou (2013). The gcdnet `R` package implements several loss functions, including logistic regression (Logit) and hybrid huberized SVM (HHSVM), with Elastic net penalty. For comparison, we used HHSVM and Logit loss functions. Since our method requires the number of classes to be known, we set $K = 3$ in fitting CCNSVM for data of Scenario 1 and $K = 6$ for data of Scenario 2. We also considered CCNSVM with known groups and HHSVM with Group lasso penalty with known groups which is implemented in the gglasso `R` software package, Yang et Zou (2015).

In both scenarios, we fitted the CCNSVM approaches with the CCN penalty using both similarity matrices, $\boldsymbol{\Omega} = \boldsymbol{X}\boldsymbol{X}^\top/n$ and $\boldsymbol{\Omega} = TOM$. We used $\tilde{\beta}_0 = 0$ and $\tilde{\boldsymbol{\beta}} = \mathbf{0}_p$ as initial values in step 2 of our algorithm to solve CCNSVM.

The method's performance is based on the following five quantities :

The number of non-zero elements in $\hat{\boldsymbol{\beta}}$.

The area under the curve, Fawcett (2006).

Misclassification rate computed in the test data and represents the percentage of observations misclassified.

Correct sparsity defined as $\frac{1}{p}\sum_{j=1}^{p} I_j$, where

$$I_j = \begin{cases} 1 & if \quad \hat{\beta}_j = \beta_j = 0, \\ 1 & if \quad \hat{\beta}_j \neq 0, \beta_j, \neq 0 \\ 0 & else. \end{cases}$$

The Rand Index, Rand (1971),

$$RI = 2(a+b)/n(n-1),$$

where a is the number of times a predictor belongs to the original cluster, b is the number of times a predictor belongs to another cluster different of the original cluster and n is the number of observations. The RI measures the similarity between the original and estimated clusters. A value close to 1 indicates the agreement between the original and the estimated clusters. For our method, the clusters are estimated directly by our algorithm. The methods that use known groups such as our approach and the HHSVM with Group-Lasso have the rand index equal to 1. For the remaining methods, the clusters are estimated by performing K-means on $\mathbf{X}_1\hat{\beta}_1, ..., \mathbf{X}_p\hat{\beta}_p$.

In both scenarios we randomly split the data into a training set and a test set with ratio 4 :1. Then, all methods are trained and tuned by five-fold cross-validation on the training set. Note that we performed a two-dimensional cross-validation approach to search for the best pair of (λ_1, λ_2) that incurs minimal misclassification error. The process was repeated 50 times and the average (and the standard error) of the five different quantities is reported in Tables 2.1, 2.2 and 2.3. Tables 2.1 and 2.2 show the performance of all methods for Scenario 1. Since the true values of the regression coefficients are known in Scenario 1, in Tables 2.1 and 2.2 we reported also the mean squares error of $\hat{\boldsymbol{\beta}}$ defined as $MSE = ||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}||_2$. In Table 2.3, we reported the results of Scenario 2 in order to show the utility of using the TOM matrix in the steps of our algorithms.

TABLE 2.1 Simulation results of Scenario 1. Means (and standard errors) over 50 replications are reported.

ρ	Method	MSC rate	AUC	RI	Sparsity	Num.Non-Zeros	$\|\|\hat{\beta}-\beta\|\|_2$
	CCN + L1 (TOM)	0.110(0.014)	0.970(0.019)	0.917(0.009)	0.892(0.043)	145.045(51.875)	6.908(0.026)
	CCN + SCAD (TOM)	0.11(0.013)	0.971(0.019)	0.919(0.010)	0.887(0.043)	151.76(49.206)	6.909(0.026)
	CCN + MCP (TOM)	0.113(0.016)	0.967(0.020)	0.916(0.01)	0.899(0.041)	136.92(49.522)	6.906(0.016)
	CCN + L1 (X)	0.097(0.013)	0.949(0.031)	0.919(0.010)	0.825(0.087)	222.905(88.483)	6.988(0.027)
	CCN + SCAD (X)	0.097(0.010)	0.954(0.032)	0.919(0.009)	0.808(0.073)	240.27(74.816)	6.985(0.030)
	CCN + MCP (X)	0.105(0.011)	0.948(0.033)	0.920(0.011)	0.820(0.090)	227.76(92.981)	6.978(0.037)
	HHSVM + EN	0.134(0.021)	0.933(0.075)	0.924(0.008)	0.86(0.041)	177.05(45.851)	6.895(0.015)
0.2	LOGIT + EN	0.145(0.03)	0.931(0.085)	0.922(0.008)	0.857(0.075)	179.42(89.162)	6.704(0.102)
	CCN + L1 KnownG (TOM)	0.118(0.024)	0.923(0.009)	1(0)	0.898(0.055)	137.166(68.069)	6.895(0.028)
	CCN + SCAD KnownG (TOM)	0.112(0.017)	0.923(0.009)	1(0)	0.870(0.048)	174.566(51.986)	6.892(0.016)
	CCN + MCP KnownG(TOM)	0.118(0.021)	0.923(0.009)	1(0)	0.894(0.045)	145.6(53.432)	6.889(0.022)
	CCN + L1 KnownG (X)	0.098(0.015)	0.906(0.009)	1(0)	0.808(0.110)	239.933(112.526)	6.958(0.047)
	CCN + SCAD KnownG (X)	0.09(0.018)	0.882(0.012)	1(0)	0.830(0.106)	217.166(108.646)	6.960(0.051)
	CCN + MCP KnownG (X)	0.098(0.015)	0.873(0.009)	1(0)	0.801(0.105)	248.5(105.938)	6.968(0.043)
	HHSVM+GL	0.157(0.019)	0.828(0.007)	1(0)	0.074(0.004)	975.833(4.564)	6.826(0.008)
	CCN + L1 (TOM)	0.056(0.010)	0.968(0.031)	0.917(0.008)	0.946(0.014)	96.400(17.662)	6.771(0.023)
	CCN + SCAD (TOM)	0.068(0.014)	0.956(0.138)	0.898(0.130)	0.716(0.340)	91.53(30.056)	6.798(0.022)
	CCN + MCP (TOM)	0.053(0.010)	0.978(0.022)	0.919(0.010)	0.906(0.029)	133.11(35.541)	6.909(0.014)
	CCN + L1 (X)	0.049(0.009)	0.973(0.016)	0.913(0.01)	0.862(0.072)	186.395(73.199)	6.977(0.036)
	CCN + SCAD (X)	0.050(0.008)	0.973(0.015)	0.911(0.008)	0.854(0.069)	195.27(70.241)	6.976(0.034)
	CCN + MCP (X)	0.050(0.011)	0.971(0.015)	0.911(0.008)	0.814(0.092)	238.575(92.087)	6.975(0.036)
	HHSVM + EN	0.066(0.017)	0.968(0.037)	0.918(0.011)	0.904(0.033)	132.27(47.649)	6.894(0.026)
0.5	LOGIT + EN	0.068(0.011)	0.940(0.144)	0.897(0.13)	0.868(0.138)	149.415(75.527)	6.585(0.031)
	CCN + L1 KnownG (TOM)	0.057(0.018)	0.978(0.006)	1(0)	0.902(0.043)	136.3(59.403)	6.889(0.017)
	CCN + SCAD KnownG(TOM)	0.055(0.013)	0.978(0.006)	1(0)	0.894(0.041)	153.367(44.943)	6.898(0.015)
	CCN + MCP KnownG(TOM)	0.059(0.016)	0.978(0.006)	1(0)	0.914(0.034)	130.3(40.357)	6.888(0.019)
	CCN + L1 KnownG (X)	0.043(0.012)	0.963(0.007)	1(0)	0.86(0.08)	190.2(79.958)	6.899(0.047)
	CCN + SCAD KnownG (X)	0.043(0.013)	0.934(0.01)	1(0)	0.88(0.075)	170(74.823)	6.922(0.057)
	CCN + MCP KnownG (X)	0.043(0.011)	0.906(0.009)	1(0)	0.869(0.101)	181(100.973)	6.925(0.059)
	HHSVM+GL	0.086(0.015)	0.858(0.004)	1(0)	0.075(0)	975(0)	6.860(0.007)

Notes : In each replication, all the methods were trained and tuned by five-fold cross-validation on the training set, using the same grid of λ_1 and λ_2, then the five statistics, Misclassification rate (MSC rate), Area under the curve (AUC), the Rand Index (RI), the Correct Sparsity, the number of non-zero elements (Num. Non.Zeros), and the mean square error of the regression parameters estimate ($||\hat{\beta}-\beta||_2$) were calculated on the test set. The correlation parameter between the predictors within each group was set to $\rho = 0.2, 0.5$.

TABLE 2.2 Simulation results of Scenario 1. Means (and standard errors) over 50 replications are reported.

ρ	Method	MSC rate	Auc	RI	Sparsity	Num.Non-Zeros	$\|\|\hat{\beta}-\beta\|\|_2$
0.8	CCN + L1 (TOM)	0.04(0.008)	0.985(0.009)	0.915(0.006)	0.911(0.134)	104.385(41.971)	6.92(0.028)
	CCN + SCAD (TOM)	0.039(0.009)	0.979(0.007)	0.916(0.006)	0.926(0.135)	91.06(26.443)	6.936(0.011)
	CCN + MCP (TOM)	0.043(0.010)	0.976(0.017)	0.917(0.007)	0.902(0.134)	121.315(39.363)	6.921(0.018)
	CCN + L1 (X)	0.039(0.009)	0.982(0.014)	0.911(0.006)	0.889(0.134)	127.92(57.832)	6.942(0.049)
	CCN + SCAD (X)	0.039(0.001)	0.978(0.014)	0.912(0.007)	0.895(0.135)	134.37(70.354)	6.846(0.146)
	CCN + MCP (X)	0.037(0.009)	0.972(0.011)	0.909(0.006)	0.851(0.146)	179.72(79.725)	6.977(0.026)
	HHSVM + EN	0.043(0.012)	0.968(0.018)	0.917(0.006)	0.892(0.140)	132.71(62.332)	6.942(0.034)
	LOGIT + EN	0.045(0.014)	0.961(0.025)	0.913(0.009)	0.892(0.136)	134.46(51.093)	6.878(0.101)
	CCN + L1 KnownG (TOM)	0.041(0.013)	0.981(0.007)	1(0)	0.931(0.03)	103.2(43.441)	6.898(0.028)
	CCN + SCAD KnownG(TOM)	0.038(0.013)	0.981(0.007)	1(0)	0.918(0.039)	128.033(44.036)	6.907(0.027)
	CCN + MCP KnownG(TOM)	0.041(0.013)	0.981(0.007)	1(0)	0.929(0.036)	116.9(38.932)	6.912(0.019)
	CCN + L1 KnownG (X)	0.030(0.011)	0.973(0.006)	1(0)	0.92(0.067)	129.8(67.320)	6.903(0.051)
	CCN + SCAD KnownG (X)	0.033(0.013)	0.95(0.01)	1(0)	0.873(0.097)	176.667(97.069)	6.885(0.057)
	CCN + MCP KnownG (X)	0.033(0.013)	0.925(0.008)	1(0)	0.93(0.058)	119.833(58.563)	6.885(0.061)
	HHSVM+GL	0.066(0.012)	0.871(0.004)	1(0)	0.075(0)	975(0)	6.878(0.008)

Notes :In each replication, all the methods were trained and tuned by five-fold cross-validation on the training set, using the same grid of λ_1 and λ_2, then the five statistics, Misclassification rate (MSC rate), Area under the curve (AUC), the Rand Index (RI), the Correct Sparsity, the number of non-zero elements (Num. Non.Zeros), and the mean square error of the regression parameters estimate ($||\hat{\beta}-\beta||_2$) were calculated on the test set. The correlation parameter between the predictors within each group was set to $\rho = 0.8$.

From Tables 2.1 and 2.2 one can see that our methods perform well compared to the HHSVM and Logit methods that use the elastic net penalty. When ρ increases, the misclassification rate decreases for all methods. The CCNSVM approach with the empirical Pearson correlation matrix (i.e. with $\boldsymbol{\Omega} = \boldsymbol{X}^\top \boldsymbol{X}/n$) behaves relatively better compared to CCNSVM with the TOM matrix, especially for the prediction of the response classes. This is not surprising since the model group structures were generated based on the correlation matrix in Scenario 1. The CCNSVM approach with the TOM matrix, however, provides sparse models while maintaining similar accuracy compared to CCNSVM with the Pearson correlation matrix. On the other hand, and as expected, our methods with known groups perform well across all simulation settings. The HHSVM with Group-Lasso fails to provide sparse models and shows modest results in terms of misclassification rate estimates, compared to

our methods. All the methods show large variation (i.e. large standard errors) for the number of non-zero coefficients, through the 50 replications. Such a model-selection variability might be due in part to the cross-validation procedure. Also, all the methods suffer from large MSE due to shrinkage bias, however, such an issue does not demonstrate major impact on the methods' accuracy. Finally, the sparse term of the CCNSVM penalty does not manifest substantial differences between Lasso, SCAD and MCP penalties.

TABLE 2.3 Simulation results of Scenario 2. Means (and standard errors) over 50 replications are reported.

Method	MSC rate	AUC	RI	Sparsity	Num.Non-Zeros
CCN + L1 (TOM)	0.159(0.005)	0.853(0.026)	0.75(0.014)	0.972(0.030)	66.17(34.566)
CCN + SCAD (TOM)	0.158(0.006)	0.864(0.013)	0.748(0.013)	0.974(0.033)	76.27(37.131)
CCN + MCP (TOM)	0.162(0.006)	0.885(0.018)	0.750(0.015)	0.952(0.058)	102.08(58.308)
CCN + L1 (X)	0.158(0.005)	0.875(0.012)	0.783(0.015)	0.948(0.037)	94.6(42.418)
CCN + SCAD (X)	0.175(0.019)	0.876(0.012)	0.763(0.019)	0.951(0.059)	58.95(72.816)
CCN + MCP (X)	0.173(0.015)	0.867(0.014)	0.758(0.019)	0.955(0.043)	50.71(61.427)
HHSVM + EN	0.166(0.026)	0.885(0.129)	0.735(0.107)	0.867(0.138)	151.8(71.192)
LOGIT + EN	0.178(0.014)	0.909(0.016)	0.776(0.011)	0.840(0.074)	205.58(78.491)
CCN + L1 KnownG (TOM)	0.165(0.012)	0.893(0.028)	1(0)	0.902(0.093)	131.533(103.111)
CCN + SCAD KnownG(TOM)	0.165(0.009)	0.889(0.029)	1(0)	0.912(0.091)	119.567(101.343)
CCN + MCP KnownG(TOM)	0.166(0.012)	0.887(0.025)	1(0)	0.931(0.065)	103.933(72.356)
CCN + L1 KnownG (X)	0.162(0.009)	0.881(0.020)	1(0)	0.925(0.101)	118.467(105.825)
CCN + SCAD KnownG (X)	0.162(0.007)	0.885(0.019)	1(0)	0.931(0.088)	110.467(94.269)
CCN + MCP KnownG (X)	0.164(0.008)	0.873(0.034)	1(0)	0.902(0.117)	141.433(122.150)
HHSVM+GL	0.166(0.013)	0.9(0.051)	1(0)	0.776(0.394)	274.167(393.672)

Notes :In each replication, all the methods were trained and tuned by five-fold cross-validation on the training set, using the same grid of λ_1 and λ_2, then the five statistics, Misclassification rate (MSC rate), Area under the curve (AUC), the Rand Index (RI), the Correct Sparsity, and the number of non-zero elements (Num. Non.Zeros) were calculated on the test set.

From Table 2.3 one can see that our approach outperforms the HHSVM and the Logit loss functions with elastic net penalty and HHSVM with the Group Lasso penalty. In this scenario CCNSVM with the TOM matrix reached accurate sparse models with the smallest classification rate estimate,

especially in the presence of unknown groups. CCNSVM with $\boldsymbol{\Omega} = \boldsymbol{X}^\top \boldsymbol{X}/n$ and unknown groups showed more sparsity for SCAD and MCP compared to Lasso, however, this tendency is not maintained for CCNSVM with known groups. Although in both scenarios we did not see the effect of SCAD and MCP compared to Lasso, we carried out additional simulations in absence of the CCN penalty term (i.e. $\lambda_2 = 0$) in the CCNSVM model (results not reported here), and we did notice that CCNSVM with SCAD (or MCP) has sparse solutions compared to Lasso.

Table 2.3 shows also that the non-cluster-based methods, HHSVM and Logit with the elastic net penalty, outperform the cluster-based methods in terms of the AUC statistic for data of Scenario 2. This can be due to the choice of the tuning parameters in the cross-validation procedure, which is based on the smallest value of the misclassification rate in the test set. This statistic measures the overall accuracy and is known to capture different characteristics of the model compared to the AUC statistic. Thus, the pair of (λ_1, λ_2) that conducts to the smallest value of the misclassification rate may not lead to smallest value of AUC.

2.4.3 Misspecification of the number of groups

Since in practice neither the presence of predictors structure nor the true value of K are unknown, we conducted a sensitivity analysis of CCNSVM in order to evaluate the impact of misspecification of the true value of K. We used the simulation set-up of Scenario 1, with two values of $\rho = \{0, 0.5\}$. The setting with value of $\rho = 0$ represents the extreme case where the design matrix has no group structure. In this case, we aimed to evaluate the performance of CCNSVM, which enforces the cluster structure between predictors, compared to the non-cluster-based methods, Logit and SVM with Elastic-Net penalty. For the case of $\rho = 0.5$, the true number of groups is $K = 3$. In this setting, we aimed to evaluate the impact of the misspecification of the true value of K for CCNSVM. In both cases, we fitted the CCNSVM approach with the Lasso penalty and we used TOM as similarity matrix. We calculated the five statistics for four different values of $K = \{2, 3, 5, 7\}$. The results comparison is based on 50 runs and reported in Tables 2.4 and 2.5, respectively.

Table 2.4 shows results when there is no group structure in the designs matrix (i.e. $\rho = 0$). From this table one can see that all the methods provide similar results. Thus, the group structure enforcement imposed by the CCNSVM approach has no impact on the method performance compared to the

non-cluster-based methods.

Table 2.5 highlights the CCNSVM results for the simulation Scenario 1 when $\rho = 0.5$. Again, one can notice from this table that the CCNSVM results are quite similar for the four different values of K although CCNSVM provides best results for $K = 3$.

TABLE 2.4 Results of the Misspecification simulation scenario (Scenario 1 with $\rho = 0$). Means (and standard errors) over 50 replications are reported.

Method	# Clusters	MSC rate	AUC	RI	Sparsity	Num.Non-Zeros
	K = 2	0.353(0.034)	0.946(0.07)	0.904(0.001)	0.893(0.043)	103.98(54.14)
CCN + L1	K = 3	0.355(0.034)	0.945 (0.067)	0.905(0.002)	0.899(0.036)	95.68(46.26)
	K =5	0.358(0.027)	0.934 (0.071)	0.905(0.003)	0.898(0.04)	95.88(53.56)
	K =7	0.352(0.035)	0.941 (0.056)	0.903(0.002)	0.904(0.039	89.68(50.18)
HHSVM + EN	—	0.38(0.074)	0.889(0.179)	0.904(0.000)	0.881(0.064)	112.42(85.93)
Logit + EN	—	0.365 (0.057)	0.904(0.149)	0.904(0.000)	0.853(0.089)	144.76(110.51)

Notes :In each replication, the methods were trained and tuned by five-fold cross-validation on the training set, using the same grid of λ_1 and λ_2, then the five statistics were calculated on the test set. For different values of $K = \{2, 3, 5, 7\}$, the CCNSVM models were fit with CCN + L1 penalty and TOM is used as similarity matrix. The second column denotes the number of clusters (# Clusters) used by CCNSVM. The remaining column labels are as in Table 2.1.

TABLE 2.5 Results of the Misspecification simulation scenario (Scenario 1 with $\rho = 0.5$). Means (and standard errors) over 50 replications are reported.

Num. Clusters	MSC rate	AUC	RI	Sparsity	Num. Non-Zeros
K=2	0.08(0.016)	0.952(0.028)	0.908(0.005)	0.934(0.033)	104.72(38.852)
K=3	0.056(0.010)	0.968(0.031)	0.917(0.008)	0.946(0.014)	96.400(17.662)
K=5	0.056(0.009)	0.975(0.015)	0.921(0.006)	0.940(0.019)	105.62(21.421)
K=7	0.057(0.011)	0.979(0.014)	0.920(0.006)	0.936(0.030)	108.44(34.139)

Notes :In each replication, the CCNSVM models were trained and tuned by five-fold cross-validation on the training set, using the same grid of λ_1 and λ_2, then the five statistics were calculated on the test set. The CCNSVM models were fit with CCN + L1 penalty and TOM is used as similarity matrix, for different values of $K = \{2, 3, 5, 7\}$. Column labels are as in Table 2.1.

2.5 Analysis of bisulfite DNA methylation sequencing data

We demonstrated the use of CCNSVM approach for classification/regression of DNA methylation data collected around the *BLK* gene located in chromosome 8 to detect differentially methylated regions (DMRs, refer to genomic regions with significantly different methylation level between two groups of samples, e.g. : cases-controls). The dataset contains measurements of DNA methylation levels of 40 different samples of cell types : B cells (8 samples), T cells (19 samples), and monocytes (13 samples). Methylation levels are measured in, $p = 5896$, CpG sites (predictors). These samples are derived from whole blood collected on a cohort of healthy individuals from Sweden. The *BLK* gene is known to have a role in B-cell receptor signalling and B-cell development, and this genomic region we have analyzed is known to be hypomethylated near the *BLK* gene in B-cells, compared to other cell types, Hertz et Stormo (1999).

We applied CCNSVM for DNA methylation classification/regression problem. We first coded the cell types as $y = \{0, 1\}$ variable, with $y = 1$ corresponds to B-cells and $y = 0$ corresponds to T- and Monocyte-cell types. The $\{0, 1\}$ response is then fitted by HHSVM and Logit loss functions with elastic net penalty, HHSVM with Group-Lasso penalty and the CCNSVM approach.

To apply our method we need to specify the number of clusters, K, for the DNA methylation data.

To determine the optimal number of clusters, we used the GAP statistic (Tibshirani *et al.* (2001) Tibshirani et al. 2001) with the Ward method ; we found $K = 11$. Since the group-Lasso approach needs known groups in order to compute the solution path, we performed a hierarchical clustering with $K = 11$ clusters and used a resulting clustering as groups in group penalized HHSVM.

The aim of this analysis is to test the performance of our methods when detecting the group of CpG sites belonging to the *BLK* gene as a DMR for the 0/1 response, and to test the power of CCNSVM in classification.

In Figure 2.2, the x-axis and the y-axis correspond respectively to the genomic position of the CpGs and the coefficient values of the optimal solutions chosen by 5-fold CV. As we can see, HHSVM with elastic net showed poor performance and the Logit with elastic net behaved better. In contrast, the region around 11.3 Mb with size 150kilo-base (kb) is detected/selected by the CCNSVM methods and HHSVM + GL method. However, we note that many points deviate slightly from the right border for the HHSVM-group-Lasso method and no points for the CCNSVM approaches, respectively. The selected genomic region is known as a DMR between DNA methylation profiles of B-cells and T/Mono cells (Turgeon, Maxime and Oualkacha, Karim and Ciampi, Antonio and Miftah, Hanane and Dehghan, Golsa and Zanke, Brent W and Benedet, Andréa L and Rosa-Neto, Pedro and Greenwood, Celia MT and Labbe, Aurélie and others (2018). These results are also in agreement with analysis conducted in (Lakhal-Chaieb *et al.* (2017)).

In a second analysis of this DNA methylation data, we randomly divided the data into the training sample of 32 observations with the remainder making up the test data. All models are fitted to the training data and the misclassification rates are calculated on test data. The tuning parameters are selected by 5-fold CV on training data. Table 2.6 summarizes the results of the average (and standard error) of the misclassification rate and the AUC statistics over 10 runs of this analysis. Table 2.6 shows that CCNSVM outperforms all its competitors in terms of distinguishing the B-cells versus T/Mono cells, for both the misclassification rate and AUC statistics.

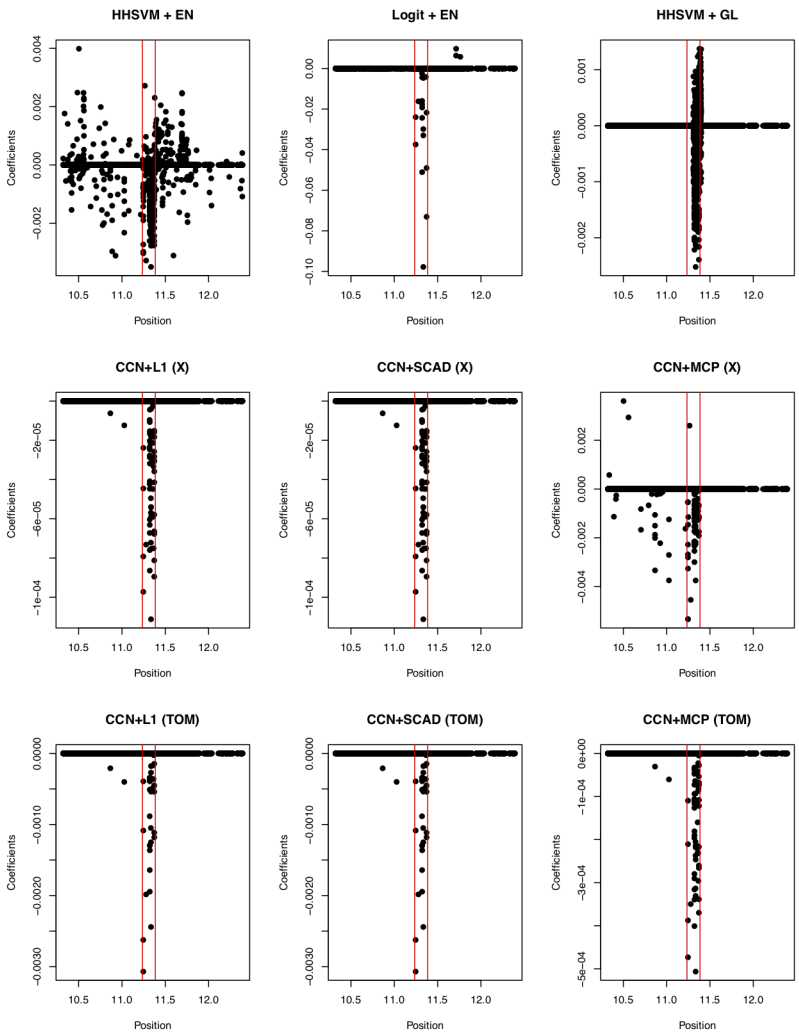

FIGURE 2.2 Comparison results of the coefficients' estimates of the 5896 CpGs sites (predictors) for the DNA methylation data analysis. For each method, the estimates are obtained as the optimal solutions chosen by five-fold CV. *Top panels* (from the left to the right) reported is the solution of the HHSVM loss function with Elastic Net penalty, Logit loss function method with Elastic Net penalty, and HHSVM with group Lasso penalty. *Mid panels* (from the left to the right) reported is the solution of CCNSVM with CCN+L1, CCNSVM with CCN+SCAD, and CCNSVM with CCN+MCP penalties ; the correlation matrix is used as a similarity matrix between the predictors. *Bottom panels* (from the left to the right) CCNSVM with CCN+L1, CCNSVM with CCN+SCAD, and CCNSVM with CCN+MCP penalties ; TOM is used as a similarity matrix between the predictors.

Method	MSC rate	AUC
CCN + L1 (TOM)	0.050(0.000)	0.992(0.011)
CCN + SCAD (TOM)	0.050(0.000)	0.992(0.011)
CCN + MCP (TOM)	0.057(0.024)	0.992(0.011)
CCN + L1 (X)	0.050(0.000)	0.984(0.000)
CCN + SCAD (X)	0.050(0.000)	0.984(0.000)
CCN + MCP (X)	0.056 (0.000)	0.975(0.000)
HHSVM + EN	0.060(0.024)	0.997(0.004))
Logit + EN	0.0575(0.016)	0.928(0.101)
HHSVM + GL	0.057(0.023)	0.966(0.025)

TABLE 2.6 Reported is the average (and standard error) of the misclassification error (MSC rate) and the area under the curve (AUC) from 10 independent splits of the DNA methylation data. All methods were fit using a two dimensional five-fold cross-validation using the same grid of λ_1 and λ_2.

2.6 Discussion

In this paper we have focused on binary classification with high-dimensional data where the number of variables p is much larger than the sample size n. We have proposed a unified one-step penalized SVM framework, which uses a Cluster Correlation-Network (CCN) penalty in combination with the Lasso, SCAD or MCP penalty. The CCN penalty is a function of the well-known Topological Overlap Matrix (TOM) whose general term measures the strength of connectivity between two predictors. We have called this approach the Cluster Correlation-Network SVM (CCNSVM). The CCNSVM framework uses the CCN penalty in order to cluster predictors in groups that are relevant to the classification variable and perform variable selection simultaneously. This is achieved via an implementation of an efficient algorithm which iterates between two steps until convergence. Regarding the first step, a search of clustering the predictors into K groups is first obtained, then step two consists of optimizing a penalized SVM function with fixed groups from the first step using Majorization-Minimization tricks together with a coordinate descent algorithm. This combination of clustering and sparsity in a single procedure provides additional insights of the power of exploring dimension reduction structure in high-dimensional binary classification. We compared the CCNSVM with other methods on bisulfite sequencing DNA methylation data and the approach achieved promising results on both classification and variable selection.

The CCNSVM algorithm has been implemented in an R package which is publicly available from

the authors via github : `github\OulkachaKarim\CCNSVM`.

2.7 Appendice **A**

Proof of Proposition 2.1

The objective function is defined by

$$Q(\beta_j|\tilde{\beta}_0,\tilde{\beta}_j)=\frac{\sum_{i=1}^n\phi_\delta(r_i)}{n}+\frac{\sum_{i=1}^n\phi'_\delta(r_i)y_ix_{ij}(\beta_j-\tilde{\beta}_j)}{n}+\frac{(\beta_j-\tilde{\beta}_j)^2}{\delta}+p_{\lambda_1,\lambda_2}(\beta_j), \tag{2.15}$$

with

$$p_{\lambda_1,\lambda_2}(\beta_j)=p_{\lambda_1}(\beta_j)+\frac{\lambda_2}{2}\sum_{k=1}^K\frac{1}{|C_k|}\sum_{j,l\in C_k}||\boldsymbol{L}_j\beta_j-\boldsymbol{L}_\ell\beta_\ell||^2.$$

The penalty $p_{\lambda_1}(\beta_j)$ takes three forms : Lasso, Mcp and Scad.

•1) $p_{\lambda_1}(\beta_j)$ is the Scad penalty :

Since, the function in the equation 2.15 is differentiable for $\beta_j\neq 0$ then we have

$$\frac{\partial Q(\beta_j|\tilde{\beta}_0,\tilde{\beta}_j)}{\partial\beta_j}=\frac{\sum_{i=1}^n\phi'_\delta(r_i)y_ix_{ij}}{n}+\frac{2(\beta_j-\tilde{\beta}_j)}{\delta}+\frac{\partial p_{\lambda_1,\lambda_2}(\beta_j)}{\partial\beta_j}. \tag{2.16}$$

Case 1 : If $|\beta_j|\leq\lambda_1$, then we have

$$\frac{\partial p_{\lambda_1,\lambda_2}(\beta_j)}{\partial\beta_j}=\lambda_1 sign(\beta_j)+\frac{\lambda_2}{|C_k|}\{(|C_k|-1)\beta_j-\sum_{\ell\neq j}\Omega_{j\ell}\tilde{\beta}_\ell\}.$$

However, since we have

$$\frac{\partial Q(\beta_j|\tilde{\beta}_0,\tilde{\beta}_j)}{\partial\beta_j}|_{\beta_j=\hat{\beta}}=0,$$

one can write

$$\{\frac{2}{\delta}+\frac{\lambda_2(|C_k|-1)}{|C_k|}\}\hat{\beta}_j=(\frac{2}{\delta}\tilde{\beta}_j+\frac{\lambda_2\sum_{\ell\neq j}\Omega_{j\ell}\tilde{\beta}_\ell}{|C_k|}-\frac{\sum_{i=1}^n\phi'_\delta(r_i)y_ix_{ij}}{n})-\lambda_1 sign(\beta_j).$$

Thus, we have to discuss two cases :

i) if $-\lambda_1 \leq \beta_j < 0$, then $sign(\beta_j) = -1$. This implies that

$$\hat{\beta}_j = \frac{Z_j + \lambda_1}{\omega} \quad \text{and} \quad -\lambda(1+\omega) \leq Z_j < -\lambda_1,$$

where $w = \frac{2}{\delta} + \frac{\lambda_2(|C_k|-1)}{|C_k|}$ and $Z_j = \frac{2}{\delta}\tilde{\beta}_j + \frac{\lambda_2 \sum_{l \neq j} \Omega_{j\ell}\tilde{\beta}_l}{|C_k|} - \frac{\sum_{i=1}^n \phi'_\delta(r_i) y_i x_{ij}}{n}$.

ii) if $0 < \beta_j \leq \lambda_1$, then $sign(\beta_j) = 1$. This implies that

$$\hat{\beta}_j = \frac{Z_j - \lambda_1}{\omega} \quad \text{and} \quad \lambda_1 < Z_j \leq \lambda_1(1+\omega).$$

In summary, we have

$$\hat{\beta}_j = \frac{S(Z_j, \lambda_1)}{\omega} \quad \text{if} \quad |\beta_j| \leq \lambda_1(1+\omega),$$

where $S(.,.)$ is the soft thresholding function defined after (2.13).

Case 2 : If $\lambda_1 < |\beta_j| \leq a\lambda_1$, then one has

$$\frac{\partial P(\beta_j)}{\partial \beta_j} = -\frac{\beta_j - a\lambda_1 sign(\beta_j)}{a-1} + \frac{\lambda_2}{|C_k|}\{(|C_k|-1)\beta_j - \sum_{\ell \neq j} \Omega_{j\ell}\tilde{\beta}_\ell\}.$$

Solving for β_j leads to

$$\hat{\beta}_j\{\omega - \frac{1}{a-1}\} = Z_j - \frac{a\lambda_1}{a-1} sign(\beta_j).$$

Again, we have to discuss to cases :

i) if $-a\lambda_1 \leq \beta_j < -\lambda$, then $sign(\beta_j) = -1$. This implies that

$\hat{\beta}_j = \frac{1}{\omega - \frac{1}{a-1}}(Z_j + \frac{a}{a-1}\lambda_1)$ if $-a\lambda_1\omega \leq Z_j < -\lambda(\omega+1)$.

ii) if $\lambda_1 < \beta_j \leq a\lambda_1$, then $sign(\beta_j) = 1$. This implies that

$\hat{\beta}_j = \frac{1}{\omega - \frac{1}{a-1}}(Z_j - \frac{a}{a-1}\lambda_1)$ if $\lambda_1(\omega+1) < Z_j \leq a\lambda_1\omega$.

In summary, one has

$$\hat{\beta}_j = \frac{S(Z_j, \frac{a\lambda_1}{a-1})}{\omega - \frac{1}{a-1}} \quad \text{if} \quad \lambda_1(\omega+1) < |Z_j| \leq a\lambda_1\omega.$$

Case 3 : If $|\beta_j| > a\lambda_1$, then we have

$$\frac{\partial P(\beta_j)}{\partial \beta_j} = \frac{\lambda_2}{|C_k|}\{(|C_k|-1)\beta_j - \sum_{\ell \neq j}\Omega_{j\ell}\tilde{\beta}_\ell\}.$$

Thus, solving for β_j leads to $\hat{\beta}_j = \frac{Z_j}{\omega}$ if $|Z_j| > a\lambda_1\omega$.

•2) $p_{\lambda_1}(\beta_j)$ is the Lasso penalty.

Following similar arguments as in the SCAD penalty developments, one can verify that $\hat{\beta}_j$ is the solution to the following equation

$$\{\frac{2}{\delta} + \frac{\lambda_2(|C_k|-1)}{|C_k|}\}\hat{\beta}_j = (\frac{2}{\delta}\tilde{\beta}_j + \frac{\lambda_2\sum_{\ell\neq j}\Omega_{j\ell}\tilde{\beta}_\ell}{|C_k|} - \frac{\sum_{i=1}^n \phi'_\delta(r_i)y_i x_{ij}}{n}) - \lambda_1 sign(\beta_j).$$

We have to discuss two cases :

Case 1 : If $\beta_j > 0$, then $sign(\beta_j) = 1$. This leads to

$$\hat{\beta}_j = \frac{Z_j - \lambda_1}{\omega}.$$

Case 2 : If $\beta_j \leq 0$, then $sign(\beta_j) = -1$. This leads to

$$\hat{\beta}_j = \frac{Z_j + \lambda_1}{\omega}.$$

In summary, we have

$$\hat{\beta}_j = \frac{S(Z_j, \lambda_1)}{\omega}.$$

3• $p_{\lambda_1}(\beta_j)$ is the MCP penalty defined by

$$P_{\lambda_1}^{MCP}(\beta_j) = \lambda_1(|\beta_j| - \frac{|\beta_j|^2}{2\lambda_1\gamma})\mathbb{1}_{|\beta_j|<\lambda_1\gamma} + \frac{\lambda_1^2\gamma}{2}\mathbb{1}_{|\beta_j|\geq\lambda_1\gamma}, \tag{2.17}$$

with $\gamma > 1$.

Case 1 : If $|\beta_j| < \lambda_1\gamma$, then following similar arguments as for SCAD penalty developments, $\hat{\beta}_j$ is the solution to the following equation

$$\{\frac{2}{\delta}+\frac{\lambda_2(|C_k|-1)}{|C_k|}-\frac{1}{\gamma}\}\hat{\beta}_j=(\frac{2}{\delta}\tilde{\beta}_j+\frac{\lambda_2\sum_{\ell\neq j}\Omega_{j\ell}\tilde{\beta}_\ell}{|C_k|}-\frac{\sum_{i=1}^n\phi'_\delta(r_i)y_ix_{ij}}{n})-\lambda_1 sign(\beta_j).$$

Here again we have to discuss two cases :

i) if $-\lambda_1\gamma<\beta_j<0$, then $sign(\beta_j)=-1$. This implies that

$\hat{\beta}_j=\frac{Z_j+\lambda_1}{\omega-\frac{1}{\gamma}}$ and $-\lambda_1\gamma\omega<Z_j<-\lambda_1$.

ii) if $0<\beta_j<\lambda_1\gamma$, then $sign(\beta_j)=1$. This implies that

$\hat{\beta}_j=\frac{Z_j-\lambda_1}{\omega-\frac{1}{\gamma}}$ and $\lambda_1<Z_j<\lambda_1\gamma\omega$.

In summary, we have

$$\hat{\beta}_j=\frac{S(Z_j,\lambda_1)}{\omega}\quad\text{if}\quad|Z_j|<\lambda_1\gamma\omega.$$

Case 2 : If $|\beta_j|>\lambda_1\gamma$, then solving for β_j leads to

$\hat{\beta}_j=\frac{Z_j}{\omega}$ if $|Z_j|>\lambda_1\gamma\omega$.

This ends the proof of Proposition 1 ■

CHAPITRE 3
HIGH-DIMENSIONAL PENALIZED BERNSTEIN SUPPORT VECTOR MACHINES

Les séparateurs à vastes marges (SVM) sont des classificateurs efficaces, très utilisés pour le classement binaire afin d'améliorer la précision de la prédiction. Cependant, la non-différenciabilité de la fonction de perte SVM peut entraîner des difficultés de calcul en grande dimension. Pour surmonter ce problème, nous proposons la fonction de perte BernSVM, une approximation lisse de la fonction de perte SVM, basée sur les polynômes de Bernstein. La fonction de perte BernSVM convient mieux au régime de grande dimension ($p >> n$). Comme la fonction BernSVM est de classe C^2, nous proposons deux algorithmes efficaces pour calculer la solution de la méthode BernSVM pénalisée. Le premier algorithme est fondé sur la technique de la descente par coordonnées combiné avec la technique de Maximisation-Minimisation (MM) et le second est un algorithme de type IRLS [de l'anglais Iterative Re-weighted Least Square]. Sous des hypothèses standard, nous démontrons une condition de cône et une condition de convexité forte restreinte afin d'établir une limite supérieure pour les estimateurs des paramètres du modèle de la méthode SVM pénalisée avec des pénalités convexes. En utilisant une approximation linéaire locale, nous étendons ce dernier résultat à la méthode SVM pénalisée avec des pénalités non convexes, comme SCAD et MCP. Notre limite supérieure est réalisable avec une grande probabilité et atteint un taux d'ordre $\sqrt{s\log(p)/n}$, où s est le nombre de prédicteurs actifs. Les études de simulation sont envisagées pour illustrer la précision de la prédiction de la méthode SVM par rapport à ses concurrents, et également pour comparer le temps de calcul et l'estimation des erreurs pour les deux algorithms proposés. L'analyse de trois exemples de données réelles de grandes dimensions a prouvé également la performance de la méthode proposée, comparée aux méthodes éxistantes.

Rachid Kharoubi, Abdallah Mkhadri et Karim Oualkacha

Abstract

The support vector machines (SVM) is a powerful classifier used for binary classification to improve the prediction accuracy. However, the non-differentiability of the SVM hinge loss function can lead to computational difficulties in high dimensional settings. To overcome this problem, we rely on Bernstein polynomial and propose a new smoothed version of the SVM hinge loss called the Bernstein support vector machine (BernSVM), which is suitable for the high dimension $p >> n$ regime. As the BernSVM objective loss function is of the class C^2, we propose two efficient algorithms for computing the solution of the penalized BernSVM. The first algorithm is based on coordinate descent with maximization-majorization (MM) principle and the second one is IRLS-type algorithm (iterative re-weighted least squares). Under standard assumptions, we derive a cone condition and a restricted strong convexity to establish an upper bound for the weighted Lasso BernSVM estimator. Using a local linear approximation, we extend the latter result to penalized BernSVM with non convex penalties SCAD and MCP. Our bound holds with high probability and achieves a rate of order $\sqrt{s\log(p)/n}$, where s is the number of active features. Simulation studies are considered to illustrate the prediction accuracy of BernSVM to its competitors and also to compare the performance of the two proposal algorithms in terms of computational timing and error estimation. The use of the proposed method is illustrated through analysis of three large-scale real data examples.

Keywords : SVM, Classification, Bernstein polynomial, Variables Selection, Non asymptotic Error Bound.

3.1 Introduction

The SVM method Cortes et Vapnik (1995) is a powerful classifier used for binary classification/ prediction. Its motivation comes from geometric considerations as it seeks a hyperplane that separates two classes of data points by the largest margins (i. e., minimal distances of observations to a hyperplane). The classification rule depends only on a subset of observations, called support vectors, which lie along the lines indicating the width of the margin. It consists of classifying a test observation based on which side of the maximal margin hyperplane it lies. However, SVM can be less efficient in high dimensional setting with a large number of variables with only few of them are relevant for classification. Nowadays, problems with sparse scale data are common in applications such as finance, document classification, image analysis and gene expression analysis. Many sparse regularized SVM approaches are proposed to control the sparsity of data and to achieve both variable selection and classification. To list a few, SVM with ℓ_1 penalty (Bradley et Mangasarian (1998) ; Zhu *et al.* (2003)), SVM with elastic net Wang *et al.* (2006), SVM with the SCAD penalty, and SVM with the combination of SCAD and ℓ_2 penalty Becker *et al.* (2011).

The objective function of these regularized SVM approaches is not differentiable at 1, so standard optimization techniques cannot be directly applied. To overcome this problem, recent alternatives are developed based on the main idea of approximating the hinge loss function with a modified smooth function, which is differentiable. Then, the use of maximization-minimization (MM) principle together with a coordinate descent algorithm (CDA) can be employed efficiently to update each component of the vector parameter of the SVM model. This idea was recently implemented in the generalized coordinate descent algorithm (gCDA) by Yang et Zou (2013) and the SVM cluster correlation-network by Kharoubi *et al.* (2019). Both authors considered the Huber loss function as a smooth approximation of the hinge loss function. They used the fact that the Huber loss is differentiable with a Lipschitz first derivative to build an upper-bound quadratic surrogate function for the Huber coordinate-wise objective function. Then, they solved the corresponding problem by a gCDA algorithm using the MM principle to ensure the descent property. Other smooth approximations of the hinge loss can be found in the ℓ_2 loss linear SVM of Chang et Chen (2008) and Lee et Mangasarian (2001). Another algorithm that is known to be computationally efficient and might be adapted to solve penalized SVM is the iterative re-weighted least squares (IRLS-type) algorithm, which solves the regularized logistic regression Friedman *et al.* (2010). However, all the objective

functions of the aforementioned methods are not twice differentiable, and thus, efficient IRLS-type algorithms can not be designed to solve such approximate sparse SVM problems.

On the other side, some works on theoretical guaranties of sparse SVM (or ℓ_1 SVM) are recently proposed, and are essentially based on the assumption that the Hessian of the SVM theoretical (expected) hinge loss function is well-defined and continuous. For example, Koo *et al.* (2008) studied the theoretical SVM loss function to overcome the differentiability of the hinge loss. Then, some appropriate assumptions about the distribution of the data $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}, i = 1, ..., n$, were considered to ensure the existence and the continuity of the Hessian with respect to the vector of coefficients of the SVM regression model. Koo *et al.* (2008) established asymptotic properties of the coefficients estimation in low-dimension ($p < n$). Zhang *et al.* (2016) used the same assumptions to establish the variable selection consistency in high-dimensions when the true vector of coefficients is sparse. Dedieu (2019) exploited the same assumptions on the theoretical hinge loss in order to develop an upper bound of the error estimation of sparse SVM in high-dimension regime.

Motivated by the computation problem of the SVM hinge loss and the discontinuity of the second derivative of the Hessian of Huber hinge loss, we propose the BernSVM loss function, which is based on the Bernstein polynomial approximation. Its second derivative is well-defined, continuous and bounded, which simplifies the implementation of a gCDA algorithm with strict descent property and also the implementation of an IRLS-type algorithm to solve penalized SVM. By construction, the proposed BernSVM loss function has suitable properties, which allows its Hessian to satisfy a restricted strong convexity Negahban *et al.* (2009). This is essential for non asymptotic theoretical guaranties developed in this work. More precisely and contrary to the works cited earlier, we consider the empirical loss function to establish an upper bound of the ℓ_2 norm of the estimation error of the BernSVM weighted Lasso estimator. Similar to Peng *et al.* (2016), we achieve an estimation error rate of order $\sqrt{s \log(p)/n}$.

This paper is organized as follows. In Section 3.2, we give more details about the construction of the BernSVM loss function using the Bernstein polynomials. Some properties of this function are oulined. Then we describe, in the same section, the two algorithms to solve the solution path of the penalized BernSVM. The first algorithm is a gCDA and the second one is an IRLS type algorithm.

In Section 3.3, we undertake the theoretical properties of the penalized BernSVM with weighetd lasso penalty. In particular, we derive an upper bound of the ℓ_2 norm of the error estimation. Then, we extend this result to penalized BernSVM with a non-convex penalty (SCAD or MPC), using local linear approximation (LLA) algorithm. The empirical performance of BernSVM based on simulation studies and application to real datasets is detailed in Section 3.4. Finally, a discussion and some conclusions are given in Section 3.5.

3.2 The penalized BernSVM

In this section, we consider the penalized BernSVM in high dimension for binary classification with convex and non-convex penalties. We first present the BernSVM loss function, which is a spline polynomial of degree fourth, as a smooth approximation to the hinge loss function. Then, we present two efficient algorithms for computing the path solution of the penalized BernSVM problem.

Assume we observe a training data of n pairs, $\{(y_1, \boldsymbol{x}_1), \ldots, (y_n, \boldsymbol{x}_n)\}$, where $\boldsymbol{x}_i \in I\!R^p$ and $y_i \in \{-1, 1\}$ denotes class labels for $i = 1, \ldots, n$.

3.2.1 The fourth degree approximation spline

We consider the problem of smoothing the SVM loss function $v : t \mapsto (1-t)_+$. The idea is to fix some $\delta > 0$, and to construct the simplest polynomial spline

$$B_\delta(t) = \begin{cases} v(t)\ , & \text{if } |t-1| > \delta \\ g_\delta(t)\ , & \text{if } |t-1| \leq \delta, \end{cases} \tag{3.1}$$

where g_δ is a decreasing polynomial, convex, and such that B_δ is twice continuously differentiable. We have in particular $B_\delta(t) \to v(t)$, as $\delta \to 0$, for all $t \in \mathbb{R}$. These conditions can be rewritten in terms of the polynomial g_δ alone as

C1. $g_\delta(1-\delta) = \delta$, $g_\delta^{'}(1-\delta) = -1$, and $g_\delta^{''}(1-\delta) = 0$;

C2. $g_\delta(1+\delta) = 0$, $g_\delta^{'}(1+\delta) = 0$, and $g_\delta^{''}(1+\delta) = 0$;

C3. $g_\delta^{''} \geq 0$.

Consider the affine transformation $t \mapsto (t-1+\delta)/2\delta$, which maps the interval $[1-\delta,\ 1+\delta]$ on $[0,\ 1]$, and let

$$q_\delta(x) = g_\delta(2\delta x + 1 - \delta)\ ,\ x \in\ [0,\ 1].$$

In terms of q_δ, the three conditions above become

C1'. $q_\delta(0) = \delta$, $q_\delta^{'}(0) = -2\delta$, and $q_\delta^{''}(0) = 0$;

C2'. $q_\delta(1) = 0$, $q_\delta^{'}(1) = 0$, and $q_\delta^{''}(1) = 0$;

C3'. $q_\delta^{''} \geq 0$.

These conditions can be dealt with in a simple manner in terms of a Bernstein basis.

3.2.1.1 The BernSVM loss function

The members of the Bernstein basis of degree m, $m \geq 0$, are the polynomials

$$b_{k,m}(x) = \begin{pmatrix} m \\ k \end{pmatrix} x^k (1-x)^{m-k},\ x \in\ [0,\ 1],$$

for $0 \leq k \leq m$. The coefficients of a polynomial P in the Bernstein basis of degree m, will be denoted by $\{c(k,\ m; P)\ :\ k = 0,\ \ldots,\ m\}$, and so we have

$$P(x) = \sum_{k=0}^{m} c(k,\ m; P) b_{k,m}(x)\ ,\ x \in\ [0,\ 1].$$

Let $\triangle$ be the forward difference operator. Here, when applied to a function h of two arguments : $(k,\ m) \mapsto h(k,\ m)$, it is understood that $\triangle$ operates on the first argument :

$$\triangle h(k,\ m) = h(k+1,\ m) - h(k,\ m) \quad \text{and}$$

$$\triangle^2 h(k,\ m) = h(k+2,\ m) - 2h(k+1,\ m) + h(k,\ m), \quad \text{for all} \quad (k, m).$$

A basic fact related to the Bernstein basis is the following for $0 \leq k \leq m$, we have

$$b'_{k,m} = m(b_{k-1,m-1} - b_{k,m-1}) = -m \triangle b_{k-1,m-1} \quad \text{for} \quad m \geq 1 \quad \text{and} \quad 0 \leq k \leq m,$$

with the convention $b_{j,m} = 0$ for $j \notin \{0,\ \ldots,\ m\}$.

A useful consequence shows how derivatives of polynomials represented in the Bernstein basis act on the coefficients. Indeed, we have

(i) $c(k,\ m-1; P') = m \triangle c(k,\ m; P)$, $0 \leq k \leq m-1$, $m \geq 1$, so that

$$P^{'}(x) = m \sum_{k=0}^{m-1} \triangle c(k,\ m; P) b_{k,m-1}(x),\ x \in\ [0,\ 1],$$

(ii) $c(k,\ m-2; P) = m(m-1) \triangle^2 c(k,\ m; P)$, $0 \leq k \leq m-2, m \geq 2$, so that

$$P^{''}(x) = m(m-1) \sum_{k=0}^{m-2} \triangle^2 c(k,\ m; P) b_{k,m-2}(x),\ x \in\ [0,\ 1].$$

Then, it is easy to see that it is hopeless to find a cubic spline B_δ satisfying the constraints. If such a spline exists, its second derivative will be a polynomial spline of degree at most one, which must equal zero at the endpoints $1-\delta$ and $1+\delta$. By continuity of the second derivative, it therefore equals 0 everywhere. The first derivative is therefore a degree zero polynomial on the entire axis and must equal -1 by the first condition. This contradicts the second condition.

Theorem 3.1 *The polynomial*

$$g_\delta(t) = \frac{1}{8\delta^3}\{\frac{(1-t+\delta)^4}{2} - (1-t-\delta)(1-t+\delta)^3\},\ t \in\ [1-\delta,\ 1+\delta] \tag{3.2}$$

is the only degree $m = 4$ polynomial satisfying the three conditions C1, C2 and C3.

The proof of Theorem 3.1 is differed to Appendix **A**.

The following proposition summarizes the properties of the first and the second derivative of the BernSVM loss function $B_\delta(.)$ defined in equation 4.3.

Proposition 3.2 *For all $t \in \mathbb{R}$, we have $|B_\delta^{'}(t)| \leq 1$ and $0 \leq B_\delta^{''}(t) \leq \frac{3}{4\delta}$.*

Proof. The loss function $B_\delta(.)$ is twice continuously differentiable. Its first derivative is given by

$$B_\delta^{'}(t) = \begin{cases} v^{'}(t)\ , & \text{if } |t-1|\ > \delta, \\ g_\delta^{'}(t)\ , & \text{if } |t-1|\ \leq \delta, \end{cases} \tag{3.3}$$

where $g_\delta^{'}(t) = \frac{(1-t+\delta)^2(1-t-2\delta)}{4\delta^3}$ and $v^{'}(t) = -\mathbf{1}_{\{1-t>\delta\}}$. Its second derivative is given by

$$B_\delta^{''}(t) = \begin{cases} 0\ , & \text{if } |t-1|\ > \delta, \\ g_\delta^{''}(t)\ , & \text{if } |t-1|\ \leq \delta, \end{cases} \tag{3.4}$$

where, $g_\delta^{''}(t) = \frac{3}{4\delta^3}[\delta^2 - (1-t)^2]$. We have, $0 \leq (1-t)^2 \leq \delta^2$ then, $0 \leq \delta^2 - (1-t)^2 \leq \delta^2$. Thus, $0 \leq g_\delta^{''}(t) \leq \frac{3}{4\delta}$ and $g_\delta^{'}(t)$ is an increasing function. We have also $g_\delta^{'}(1-\delta) = -1$ and $g_\delta^{'}(1+\delta) = 0$, then, $-1 \leq p_\delta^{'}(t) \leq 0$. Thus, $\forall t \in \mathbb{R}$ we have $|B_\delta^{'}(t)| \leq 1$. □

3.2.1.2 Graphical illustration

In this section, we examine some properties of the BernSVM loss function through a graphical illustration. We also compare its behavior with the standard SVM loss function $v(t)$ and the hinge loss function considered in HHSVM Yang et Zou (2013), which is given by

$$\phi_c(t) = \begin{cases} 0 & , t \geq 1, \\ \frac{(1-t)^2}{2\delta} & , 1-\delta < t \leq 1, \\ 1 - t - \frac{\delta}{2} & , t \leq 1-\delta. \end{cases}$$

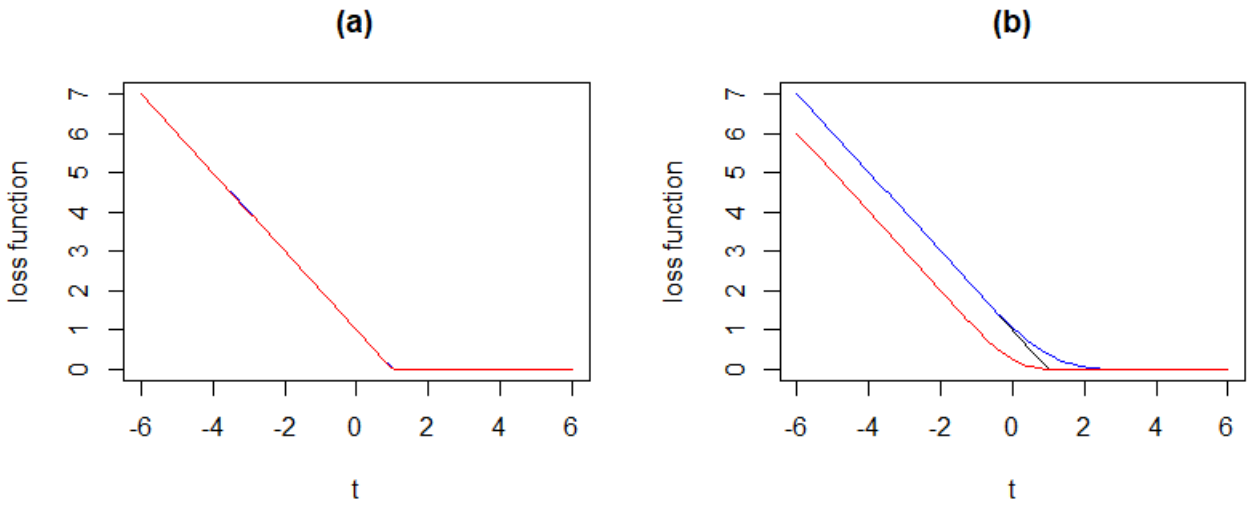

FIGURE 3.1 (a) The Huber loss (red) with $\delta = 0.01$, the BernSVM loss (blue) with $\delta = 0.01$ and the SVM loss (black), (b) The Huber loss (red) with $\delta = 2$, the BernSVM loss (blue) with $\delta = 2$ and the SVM loss (black)

Figure 3.1 shows this graphical illustration. When δ is small (i.e. close to zero), the three loss functions are almost identical (Panel (a)). When $\delta \geq 1$ (Panel (b)), both HHSVM and BernSVM loss functions have the same shape that the SVM loss. However, BernSVM approximates better the SVM loss function. Moreover, BernSVM loss function is twice differentiable everywhere.

3.2.2 Algorithms for solving penalized BernSVM

In this section, we propose the penalized BernSVM as an alternative to the penalized SVM and the penalized HHSVM for binary classification in high dimension settings. We are assuming n pairs of training data $(\boldsymbol{x}_i, y_i)$, for $i = 1, ..., n$, with $\boldsymbol{x}_i \in \mathbb{R}^p$ predictors and $y_i \in \{-1, 1\}$. We assume that the predictors are standardized :

$$\frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0, \quad \frac{1}{n}\sum_{i=1}^{n} x_{ij}^2 = 1.$$

We propose to solve the penalized BernSVM problem given by

$$\min_{(\beta_0,\boldsymbol{\beta})} \frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})) + P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}), \tag{3.5}$$

where the objective function $B_\delta(.)$ is defined in (4.3) and

$$P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) = \sum_{j=1}^{p} P_{\lambda_1}(|\beta_j|) + \frac{\lambda_2}{2}||\boldsymbol{\beta}||_2^2$$

is a penalty function added for obtaining sparse coefficients' estimates. The penalty $P_{\lambda_1}(|\beta_j|)$ takes different forms :

- If $P_{\lambda_1}(|\beta_j|) = \lambda_1|\beta_j|$, $P_{\lambda_1,\lambda_2}(\boldsymbol{\beta})$ is the elastic net penalty (EN) ;
- If $P_{\lambda_1}(|\beta_j|) = \lambda_1 w_j|\beta_j|$, where the weight $w_j > 0$ for all j, then $P_{\lambda_1,\lambda_2}(\boldsymbol{\beta})$ is the adaptive Lasso + L2 norm penalty (AEN) ;
- If one set

  $$P_{\lambda_1}(|\beta_j|) = \lambda_1|\beta_j|\mathbb{1}_{|\beta_j|\leq\lambda_1} - \frac{|\beta_j|^2 - 2a\lambda_1|\beta_j| + \lambda_1^2}{2(a-1)}\mathbb{1}_{\lambda_1<|\beta_j|\leq a\lambda_1} + \frac{(a+1)\lambda_1^2}{2}\mathbb{1}_{|\beta_j|>a\lambda_1},$$

  with $a > 2$, then $P_{\lambda_1,\lambda_2}(\boldsymbol{\beta})$ is the SCAD + L2 norm penalty (SCADEN) ;
- If one set

  $$P_{\lambda_1}(|\beta_j|) = \lambda_1(|\beta_j| - \frac{|\beta_j|^2}{2\lambda_1 a})\mathbb{1}_{|\beta_j|<\lambda_1 a} + \frac{\lambda_1^2 a}{2}\mathbb{1}_{|\beta_j|\geq\lambda_1 a},$$

  where $a > 1$, then in this case, $P_{\lambda_1,\lambda_2}(\boldsymbol{\beta})$ is the MCP + L2 norm penalty (MCPEN).

Before going through details of the proposed algorithms to solve BernSVM optimization problem, we would like to emphasize that, for small δ, the minimizer of the penalized BernSVM in (3.5) is a good approximation to the minimizer of the penalized SVM. This is outlined in the following proposition.

Proposition 3.3 *Let the penalized SVM loss function be defined as follows*

$$R(\boldsymbol{\beta}, \beta_0) = \frac{1}{n}\sum_{i=1}^{n} v(y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})) + P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}),$$

and let the penalized BernSVM loss function be given as

$$R(\boldsymbol{\beta}, \beta_0|\delta) = \frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})) + P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}).$$

Then, we have

$$\inf_{(\boldsymbol{\beta},\beta_0)} R(\boldsymbol{\beta}, \beta_0) \leq \inf_{(\boldsymbol{\beta},\beta_0)} R(\boldsymbol{\beta}, \beta_0|\delta) \leq \inf_{(\boldsymbol{\beta},\beta_0)} R(\boldsymbol{\beta}, \beta_0) + \delta.$$

Proof. We have

$$B_\delta(t) = v(t)\mathbb{1}_{(|t-1|>\delta)}(t) + g_\delta(t)\mathbb{1}_{(|t-1|\leq\delta)}(t),$$

where, $\mathbb{1}_A(t) = 1$ *if* $t \in A$ *and 0 otherwise. Then, one can write*

$$B_\delta(t) - v(t) = \begin{cases} 0\ , & \text{if } |t-1| > \delta, \\ g_\delta(t) \geq 0\ , & \text{if } |t-1| \leq \delta, \end{cases}$$

which means that $v(t) \leq B_\delta(t)$*. We have also that* $g_\delta(t)$ *is decreasing for any* $t \in [1-\delta, 1+\delta]$ *because* $g_\delta^{'}(t) \leq 0$*. Thus, one has* $g_\delta(t) \leq g_\delta(1-\delta) = \delta$*, which implies*

$$v(t) + \delta - B_\delta(t) = \begin{cases} \delta \geq 0\ , & \text{if } |t-1| > \delta, \\ \delta - g_\delta(t) \geq 0\ , & \text{if } |t-1| \leq \delta. \end{cases}$$

We conclude that

$$v(t) \leq B_\delta(t) \leq v(t) + \delta.$$

This means that

$$R(\boldsymbol{\beta}, \beta_0) \leq R(\boldsymbol{\beta}, \beta_0|\delta) \leq R(\boldsymbol{\beta}, \beta_0) + \delta,$$

which leads to

$$\inf_{(\boldsymbol{\beta},\beta_0)} R(\boldsymbol{\beta}, \beta_0) \leq \inf_{(\boldsymbol{\beta},\beta_0)} R(\boldsymbol{\beta}, \beta_0|\delta) \leq \inf_{(\boldsymbol{\beta},\beta_0)} R(\boldsymbol{\beta}, \beta_0) + \delta.$$

□

In the remaining of section 3.2, we propose two competitor algorithms for solving (3.5) with convex penalties. The first one is a gCDA algorithm based on MM principle, while the second one is an IRLS-type algorithm. Moreover, we briefly describe a third algorithm based on local linear approximation for solving (3.5) with non-convex penalties.

3.2.2.1 The coordinate descent algorithm for penalized BernSVM

In this section, we present a gCDA algorithm to solve penalized BernSVM, called BSVM-GCD, similar to the gCDA approach proposed by Yang et Zou (2013). Because the second derivative of the BernSVM objective function is bounded, we approximate the coordinate wise objective function by a surrogate function and we use the coordinate descent algorithm to solve the optimization problem in (3.5). The proposed approach is described next.

Notice, first, that the coordinate objective function of problem (3.5) can be written as follows

$$F(\beta_j|\beta_0,\tilde{\beta}_j) := \frac{1}{n}\sum_{i=1}^{n} B_\delta\{r_i + y_i x_{ij}(\beta_j - \tilde{\beta}_j)\} + P_{\lambda_1,\lambda_2}(\beta_j), \tag{3.6}$$

where $r_i = y_i(\tilde{\beta}_0 + \boldsymbol{x}_i^\top \tilde{\boldsymbol{\beta}})$, and $\tilde{\beta}_0$ and $\tilde{\boldsymbol{\beta}}$ are the current estimates of β_0 and $\boldsymbol{\beta}$, respectively. As the loss function, $B_\delta(.)$, is convex, twice differentiable and has a second derivative bounded by $L = 3/4\delta$, one can rely on the mean value theorem and approximate the objective function in (3.6) by a (surrogate) quadratic function, $Q_\delta(.)$, and then use the MM principle to solve the new problem for each β_j holding $\tilde{\beta}_0$ and $\beta_k = \tilde{\beta}_j$, for $k \neq j$, fixed at the current iteration

$$\min_{\beta_j} Q_\delta(\beta_j|\tilde{\beta}_0,\tilde{\beta}_j), \tag{3.7}$$

where

$$Q_\delta(\beta_j|\beta_0,\tilde{\beta}_j) = \frac{\sum_{i=1}^n B_\delta(r_i)}{n} + \frac{\sum_{i=1}^n B'_\delta(r_i) y_i x_{ij}}{n}(\beta_j - \tilde{\beta}_j) + \frac{L}{2}(\beta_j - \tilde{\beta}_j)^2 + P_{\lambda_1,\lambda_2}(\beta_j). \tag{3.8}$$

The next proposition gives the explicit solution of (3.7) for the different forms of $P_{\lambda_1}(\beta_j)$.

Proposition 3.4 *Let $Q_\delta(\beta_j|\tilde{\beta}_0,\tilde{\beta}_j)$ be the surrogate loss function defined in (3.8). Let $P_{\lambda_1}(.)$ be L_1-norm or adaptive Lasso penalty. The closed form solution of the minimizer of (3.7) is given as follows*

$$\hat{\beta}_j^{EN} = \frac{S(Z_j,\lambda_1)}{\omega}, \tag{3.9}$$

$$\hat{\beta}_j^{AEN} = \frac{S(Z_j,\lambda_1 w_j)}{\omega}, \tag{3.10}$$

where $S(a,b) = (|a|-b)_+ sign(a)$, $Z_j = -\frac{\sum_{i=1}^n B'_\delta(r_i) y_i x_{ij}}{n} + L\tilde{\beta}_j$, and $\omega = \lambda_2 + L$ and $L = 3/4\delta$.

The proof is outlined in Appendix **A**.

Algorithm 3 (BSVM-GCD), given next, gives details of steps for solving the objective function in (3.8) using both coordinate descent and MM principle.

Algorithm 3 The BSVM-GCD algorithm to solve the BernSVM loss function with elastic net and adaptive net penalties.

---

1. Initialize $\tilde{\beta}_0$ and $\tilde{\boldsymbol{\beta}}$;
2. Iterate the following updates until convergence :
   (a) For $j = 1, \ldots, p$, update $\tilde{\beta}_j$
   * compute $r_i = y_i(\tilde{\beta}_0 + \mathbf{x}_{i.}^\top \tilde{\boldsymbol{\beta}}) \quad for \quad i = 1, \ldots, n;$
   * set
   $$\tilde{\beta}_j^{\text{new}} \longleftarrow \hat{\beta}_j,$$
   with $\hat{\beta}_j$ is given by (3.9) for the Elastic Net or (3.10) for the Adaptive Net;
   (b) Update $\tilde{\beta}_0$
   * compute $r_i = y_i(\tilde{\beta}_0 + \mathbf{x}_{i.}^\top \tilde{\boldsymbol{\beta}}) \quad for \quad i = 1, \ldots, n;$
   * set
   $$\tilde{\beta}_0^{\text{new}} \longleftarrow \tilde{\beta}_0 - L^{-1} \frac{\sum_{i=1}^n B'_\delta(r_i) y_i}{n}.$$

The upper bound of the second derivative of the BernSVM loss function can be reached at $t = 1$. This means that $F(t|\beta_0, \tilde{\beta}_j) \leq Q(t|\beta_0, \tilde{\beta}_j)$. To reach a strict inequality, one can relax the upper bound L and use $\tilde{L} = (1 + \epsilon)L$, where $\epsilon = 1e^{-6}$. Hence, Algorithm 3 is implemented using $\tilde{L}$ in the place of L. This leads to the strict descent property of the BSVM-GCD algorithm, which is given in the next proposition.

Proposition 3.5 *The strict descent property of the BSVM-GCD approach is obtained using the upper bound $\tilde{L}$ and given by*

$$F(\beta_j|\beta_0, \tilde{\beta}_j) = Q(\beta_j|\beta_0, \tilde{\beta}_j), \quad if \quad \beta_j = \tilde{\beta}_j,$$

$$F(\beta_j|\beta_0, \tilde{\beta}_j) < Q(\beta_j|\beta_0, \tilde{\beta}_j), \quad if \quad \beta_j \neq \tilde{\beta}_j.$$

Proof. Using Taylor expansion and the fact that the BernSVM loss function is twice differentiable, we have

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n} B_\delta\{r_i + y_i x_{ij}(\beta_j - \tilde{\beta}_j)\} &= \frac{1}{n}\sum_{i=1}^{n} B_\delta(r_i) + \frac{1}{n}\sum_{i=1}^{n} y_i x_{ij} B'_\delta(r_i)(\beta_j - \tilde{\beta}_j) \\ &+ \frac{1}{2n}\sum_{i=1}^{n} y_i^2 x_{ij}^2 B''_\delta(r_i)(\beta_j - \tilde{\beta}_j)^2 \\ &\leq \frac{1}{n}\sum_{i=1}^{n} B_\delta(r_i) + \frac{1}{n}\sum_{i=1}^{n} y_i x_{ij} B'_\delta(r_i)(\beta_j - \tilde{\beta}_j) + \frac{L}{2}(\beta_j - \tilde{\beta}_j)^2,\end{aligned}$$

The last inequality holds because we have $y_i^2 = 1$, $\sum_{i=1}^{n} x_{ij}^2 = n$, *and* $B''_\delta(r_i) \leq L$ *from Proposition 1. Hence, we have* $F(\beta_j|\beta_0, \tilde{\beta}_j) = Q(\beta_j|\beta_0, \tilde{\beta}_j)$ *if* $\beta_j = \tilde{\beta}_j$, *and we have* $F(\beta_j|\beta_0, \tilde{\beta}_j) < Q(\beta_j|\beta_0, \tilde{\beta}_j)$, *for* $\beta_j \neq \tilde{\beta}_j$. □

This proposition shows that the objective function $F(.)$ decreases after each majorization update, which means that for $\tilde{\beta}_j^{new} \neq \tilde{\beta}_j$ we have

$$F(\beta_j^{new}|\beta_0, \tilde{\beta}_j) < F(\tilde{\beta}_j|\beta_0, \tilde{\beta}_j).$$

This result shows that the BSVM-GCD algorithm enjoys strict descent property.

3.2.2.2 The IRLS algorithm for penalized BernSVM

In this section, we present an IRLS-type algorithm combined with coordinate descent method, termed BSVM-IRLS, to solve the penalized BernSVM in (3.5). In their approach, Friedman *et al.* (2010) used IRLS to solve the regularized logistic regression because the second derivative exists. On the other hand, Yang et Zou (2013) used a gCDA algorithm to solve the regularized HHSVM because the Huber loss does not have the second derivative everywhere. Our BernSVM loss function, in contrast, is very smooth and thus one can rely on this advantage and solve (3.5) using IRLS trick combined with coordinate descent method. The BSVM-IRLS algorithm is described below.

Firstly, we consider the non-penalized BernSVM with loss function

$$\ell(\beta_0, \boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})) = \frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i \boldsymbol{X}_i^\top \boldsymbol{b}), \tag{3.11}$$

where $\boldsymbol{b} = (\beta_0, \boldsymbol{\beta}^\top)^\top$ and $\boldsymbol{X}_i = (1, \boldsymbol{x}_i^\top)^\top$. To solve

$$\frac{\partial \ell}{\partial \boldsymbol{b}}(\beta_0, \boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} y_i \boldsymbol{X}_i B'_\delta(y_i \boldsymbol{X}_i^\top \boldsymbol{b}) = 0,$$

one can use the Newton-Raphson (NR) algorithm because the loss function has a continuous second derivative defined by

$$\frac{\partial^2 \ell}{\partial \boldsymbol{b} \partial \boldsymbol{b}^\top}(\beta_0, \boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{X}_i B_\delta''(y_i \boldsymbol{X}_i^\top \boldsymbol{b}) \boldsymbol{X}_i^\top.$$

The update of the coefficients is given then by

$$\boldsymbol{b}^{new} = \tilde{\boldsymbol{b}} - (\frac{\partial^2 \ell}{\partial \boldsymbol{b} \partial \boldsymbol{b}^\top}(\beta_0, \boldsymbol{\beta}))^{-1} \frac{\partial \ell}{\partial \boldsymbol{b}}(\beta_0, \boldsymbol{\beta}) \tilde{\boldsymbol{b}}.$$

Let $u_i = y_i . B_\delta'(y_i \boldsymbol{X}_i^\top \tilde{\boldsymbol{b}})$, $\phi_{ii} = B_\delta''(y_i \boldsymbol{X}_i^\top \tilde{\boldsymbol{b}})$ and $\boldsymbol{\Phi} = diag(\phi_{ii})$ for $i = 1 \ldots, n$. Then, the update of the NR algorithm can be written as follows,

$$\begin{aligned} \boldsymbol{b}^{new} &= \tilde{\boldsymbol{b}} - (\boldsymbol{X}^\top \boldsymbol{\Phi} \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{u} \\ &= (\boldsymbol{X}^\top \boldsymbol{\Phi} \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{\Phi} [\boldsymbol{X} \tilde{\boldsymbol{b}} - \boldsymbol{\Phi}^{-1} \boldsymbol{u}] \\ &= (\boldsymbol{X}^\top \boldsymbol{\Phi} \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{\Phi} \boldsymbol{z}, \end{aligned}$$

where $\boldsymbol{z} = \boldsymbol{X} \tilde{\boldsymbol{b}} - \boldsymbol{\Phi}^{-1} \boldsymbol{u}$. Thus, $\boldsymbol{b}^{new}$ is the update of a weighted least squares regression problem with response $\boldsymbol{z}$, with the loss function

$$\ell(\beta_0, \boldsymbol{\beta}) \approx \frac{1}{2n} \sum_{i=1}^{n} \phi_{ii} (z_i - \boldsymbol{X}_i^\top \boldsymbol{b})^2.$$

This problem can be solved by an IRLS algorithm in the case $p < n$. In high dimensional settings, the penalized BernSVM, defined in (3.5), can be solved by combining IRLS with a cyclic coordinate descent method. Thus, for the Adaptive Elastic Net penalty, the solution is given by

$$\hat{\beta}_j^{AEN} = \frac{S(\frac{\sum_{i=1}^n \phi_{ii} x_{ij} r_i}{n} + \frac{\sum_{i=1}^n \phi_{ii} x_{ij}^2}{n} \tilde{\beta}_j, w_j \lambda_1)}{\lambda_2 + \frac{\sum_{i=1}^n \phi_{ii} x_{ij}^2}{n}} \quad \text{and} \quad \hat{\beta}_0^{AEN} = \frac{\sum_{i=1}^n \phi_{ii} r_i}{\sum_{i=1}^n \phi_{ii}}, \tag{3.12}$$

where $r_i = z_i - \tilde{\beta}_0 - \boldsymbol{x}_i^\top \tilde{\boldsymbol{\beta}}$ and $(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$ is the estimate of $(\beta_0, \boldsymbol{\beta})$ obtained on the current iteration. Of note, to deal with the zero weights (i.e., $\phi_{ii} = 0$ for some i), one can replace them by a constant $\xi = 0.001$ or use a modified NR algorithm and replace all the weights by an upper bound of the second derivative. In our work, we adopted the modified NR algorithm and set $\phi_{ii} = L$ for $i = 1, \ldots, n$, where $L = \frac{3}{4\delta}$ is the upper bound of the second derivative of the proposed loss function $B_\delta(.)$ given by (4.3).

The algorithm of solving BernSVM with IRLS algorithm is given next.

Algorithm 4 The BSVM-IRLS to solve the BernSVM with elastic net.

1. Initialize $\tilde{\boldsymbol{b}} = (\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}^\top)^\top$ and let $L = \frac{3}{4\delta}$;
2. Iterate the following updates until convergence :
   (a) Calculate $\boldsymbol{u} = (u_1, \ldots, u_n)^\top$, where $u_i = y_i.B'_\delta(y_i \boldsymbol{X}_i^\top \tilde{\boldsymbol{b}})$;
   (b) Calculate the working response $\boldsymbol{z} = \boldsymbol{X}\tilde{\boldsymbol{b}} - L^{-1}\boldsymbol{u}$;
   (c) Using cyclic coordinate descent to solve the penalized weighted Least square

$$\hat{\boldsymbol{b}} = (\hat{\beta}_0, \hat{\boldsymbol{\beta}})^\top = \underset{(\beta_0, \boldsymbol{\beta})}{arg\,\min} \ell(\beta_0, \boldsymbol{\beta}),$$

   where,$(\hat{\beta}_0, \hat{\beta}_j), j = 1, \ldots, p$, are given by (4.2) ;
   (d) $\tilde{\boldsymbol{b}} = \hat{\boldsymbol{b}}$.

3.2.2.3 Local linear approximation algorithm for BernSVM with non-convex penalties

We consider in this section the penalized BernSVM with non-convex penalty SCAD or MCP and we propose to solve the resulting optimization problem using the local linear approximation (LLA) of Zou et Li (2008).

Without loss of generality, we assume in this section that $\lambda_2 = 0$. The LLA approach is an appropriate approximation of SCAD or MCP penalty. It is based on the first order Taylor expansion of the MCP or SCAD penalty functions around $|\tilde{\beta}_j|$, and adapts both penalties to be solved as an iterative adaptive Lasso penalty ; the weights are updated/estimated at each iteration. In our context, this means that the penalized BernSVM with SCAD or MCP penalty can be solved iteratively as follows

$$(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})^{new} = argmin_{(\beta_0, \boldsymbol{\beta})} \frac{1}{n} \sum_{i=1}^{n} B_\delta(y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})) + \lambda_1 ||\hat{\boldsymbol{W}}\boldsymbol{\beta}||_1, \tag{3.13}$$

where $\hat{\boldsymbol{W}} = diag\{\hat{w}_j\}$, with $\hat{w}_j = P'_{\lambda_1}(\tilde{\beta}_j)/\lambda_1, j = 1, \ldots, p$, are the current weights, and $P'_{\lambda_1}(.)$ is the first derivative of the SCAD or MCP penalty given respectively by

$$P'_{\lambda_1}(t) = \lambda_1 \mathbb{1}_{\{t \leq \lambda_1\}} + \frac{(a\lambda_1 - t)_+}{a - 1} \mathbb{1}_{\{t > \lambda_1\}},$$

and

$$P'_{\lambda_1}(t) = (\lambda_1 - \frac{t}{a})_+.$$

To sum up, penalized BernSVM with LLA penalty is an iterative algorithm, whcih solves at each iteration BSVM-IRLS (or BSVM-GCD). The steps of the LLA algorithm for penalized BernSVM are described in Algorithm 5 below.

Algorithm 5 The local linear approximation algorithm to solve penalized BernSVM with SCAD or MCP penalty.

1. Initialize $\tilde{\beta}_0$ and $\tilde{\boldsymbol{\beta}}$ and compute $\hat{w}_j^0 = P'_{\lambda_1}(|\tilde{\beta}_j|)/\lambda_1$ for $j = 1, \ldots, p$;
2. Iterate the following updates until convergence :
   (a) For $j = 0, 1, \ldots, p$, update β_0 and β_j by solving the problem in (3.13) using either BSVM-IRLS or BSVM-GCD algorithms;
   (b) Set $\hat{w}_j^{\text{new}} = P'_{\lambda_1}(\tilde{\beta}_j^{\text{new}})/\lambda_1$.

3.3 Theoretical guaranties for penalized BernSVM estimator

In the next section, we develop an upper bound of the ℓ_2 norm of the estimation error $||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2$ for penalized BernSVM with weighted lasso ($\lambda_2 = 0$) and SCAD or MCP penalties. Here, $\hat{\boldsymbol{\beta}}$ is the minimizer of the corresponding penalized BernSVM and $\boldsymbol{\beta}^\star$ is the minimizer of the population version of (3.5) without the penalty term; i.e., $\boldsymbol{\beta}^\star$ is the minimizer of $\mathbb{E}[B_\delta(y\boldsymbol{x}^\top\boldsymbol{\beta})]$. For seek of simplicity, the intercept is omitted for the theoretical development.

3.3.1 An upper bound of the estimation error for weighted lasso penalty

In this section, we derive an upper bound of the estimation error $||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2$ for penalized BernSVM with weighted lasso penalty using the properties of the BernSVM loss function.

The weighted penalized BernSVM optimization problem is defined as follows

$$\arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} L(\boldsymbol{\beta}) := \frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i\boldsymbol{x}_i^\top\boldsymbol{\beta}) + \lambda_1||\boldsymbol{W}\boldsymbol{\beta}||_1, \tag{3.14}$$

where $\boldsymbol{W} = diag(\boldsymbol{w})$ is a diagonal matrix of known weights, with $w_j \geq 0$, for $j = 1 \ldots p$. The (unweighted) Lasso is a special case of (3.14), with $w_j = 1, j \in \{1, \ldots, p\}$.

Let $\hat{\boldsymbol{\beta}}$ be a minimizer of (3.14), and assume that the population-level minimizer $\boldsymbol{\beta}^\star$ defined above is sparse and unique. Define $S \subset \{1, 2, ..., p\}$ the set of non-zero elements of $\boldsymbol{\beta}^\star$, with $|S| = s$, and S^c is the complement of S.

In order to obtain an upper bound of the estimation error, the BernSVM loss function needs to satisfy the strongly convex assumption. This means that the minimum of the eigenvalues of its corresponding Hessian matrix, defined as $\boldsymbol{H} = \boldsymbol{X}^\top \boldsymbol{D}\boldsymbol{X}/n$, is lower bounded by a constant $\kappa > 0$; where the matrix $\boldsymbol{D} = diag\{B_\delta^{"}(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star)\}$, $i = 1, \ldots, n$. In high-dimensional settings, with $p > n$, $\boldsymbol{H}$ has rank lower or equal to n. Therefore, we can not reach the strong convexity assumption of the Hessian matrix. To overcome this problem, one can look for a restricted strong convexity assumption to be verified on a restricted set $\mathcal{C} \subset \mathbb{R}^p$ Negahban *et al.* (2009), which is defined next.

Definition 3.6 (Restricted Strong Convexity (RSC)) *A design matrix $\boldsymbol{X}$ satisfies the RSC condition on $\mathcal{C}$ with $\kappa > 0$ if*

$$\frac{||\boldsymbol{X}\boldsymbol{h}||_2^2}{n} \geq \kappa ||\boldsymbol{h}||_2^2, \quad \forall \boldsymbol{h} \in \mathcal{C}.$$

We also state the following assumptions :

A1 : Let

$$\mathcal{C}(S) = \{\boldsymbol{h} \in \mathbb{R}^p : ||\boldsymbol{W}_{\boldsymbol{S}^c} \boldsymbol{h}_{\boldsymbol{S}^c}||_1 \leq \gamma ||\boldsymbol{h}_S||_1\},$$

where $\gamma > 0$ and $\boldsymbol{h}_{\mathcal{A}} = (h_j : j \in \mathcal{A})$, with $\mathcal{A} \subset \{1, \ldots, p\}$, is a sub-vector of $\boldsymbol{h}$. We assume that the design matrix, for all $\boldsymbol{h} \in \mathcal{C}$, satisfies the condition

$$\frac{||\boldsymbol{X}\boldsymbol{h}||_2^2}{n} \geq \kappa_1 ||\boldsymbol{h}||_2^2 - \kappa_2 \frac{\log(p)}{n} ||\boldsymbol{h}||_1^2,$$

with probability $1 - \exp(-c_0 n)$, for some $c_0 > 0$, where $\boldsymbol{x}_i, i = 1, \ldots, n$, are i.i.d Gaussian or Sub-Gaussian, $\kappa_1 and \kappa_2$ positive constant's.

A2 : There exist a ball, $\boldsymbol{B}(\boldsymbol{x}_0, r_0)$, centered at a point $\boldsymbol{x}_0$, with radius r_0, such as $B_\delta^{''} \circ f(\boldsymbol{x}_0) > 0$, with $f(\boldsymbol{x}_0) = y \boldsymbol{x}_0^\top \boldsymbol{\beta}^\star$, and for any $\boldsymbol{x} \in \boldsymbol{B}(\boldsymbol{x}_0, r_0)$, we have $B_\delta^{''} \circ f(\boldsymbol{x}) > \kappa_3$, for some constant $\kappa_3 > 0$.

A3 : The columns of the design matrix $\boldsymbol{X}$ are bounded :

$$\forall j \in \{1, ..., p\} \quad ||\boldsymbol{x}_j||_2 \leq M.$$

Assumption **A1** is satisfied with high probability for Gaussian random matrices Raskutti *et al.* (2010). Rudelson et Zhou (2012) extended this result to the Sub-Gaussian random matrices. Assumption **A2** guarantees the existence of $\boldsymbol{x}_0$. In fact, for $\delta > 0$, we have $B_\delta^{''}(1) = \frac{3}{4\delta} > 0$. Thus, we

can choose $\boldsymbol{x}_0 \in g^{-1}(\{1\}) \subset \mathbb{R}^p$, where $g(\boldsymbol{x}_0) = y\boldsymbol{x}_0^\top \boldsymbol{\beta}^\star$. As the second derivative of the BernSVM loss function exists and continuous, the function $B_\delta^{''} \circ g(\boldsymbol{x}_0)$ is also continuous in $\boldsymbol{x}_0$ and satisfies $B_\delta^{''} \circ g(\boldsymbol{x}_0) > 0$. The existence of a ball around $\boldsymbol{x}_0$ as described in **A2** is a result of the continuity properties of $B_\delta^{''} \circ g(.)$.

Assumption **A2** is needed to ensure the positive-definiteness of the Hessian matrix around $\boldsymbol{\beta}^\star$ in order to derive an upper bound of the L_2 norm of $||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||$. See Lemma 1 in Appendix **A** for more detail concerning the use of this assumption. Note that in the case of studying theoretical properties of the standard SVM model, to reach a similar property for the Hessian, Koo *et al.* (2008) assumed three assumptions, namely **A1**, **A3** and **A4**; see Lemma 5 of Koo *et al.* (2008).

The next lemma establishes that the Hessian matrix, $\boldsymbol{H}$, satisfies the RSC condition with high probability.

Lemma 3.7 *Under assumptions* $\boldsymbol{A}1$ *and* $\boldsymbol{A}2$*, if*

$$n > \frac{2\kappa_2 s \log(p)(1 + \gamma||\boldsymbol{W}_{S^c}^{-1}||_\infty)^2}{\kappa_1},$$

then the Hessian matrix $\boldsymbol{H}$ *satisfies the RSC condition with* $\kappa = \frac{\kappa_3\kappa_1}{2}$ *on* $\mathcal{C}$ *with probability* $1 - \exp(-c_0 n)$*, for some universal constant* $c_0 > 0$.

The proof of Lemma 3.7 is given in Appendix **B**.

The following theorem establishes an upper bound of the penalized BernSVM with Adaptive Lasso.

Theorem 3.8 *Assume that Assumptions* $\boldsymbol{A}1 - \boldsymbol{A}3$ *are met and* $\lambda_1 \geq \frac{M(1+\gamma||\boldsymbol{W}_{\boldsymbol{S}^c}^{-1}||_\infty)}{\sqrt{n}(\gamma - ||\boldsymbol{W}_{\boldsymbol{S}}||_\infty)}$ *for any* $\gamma > ||\boldsymbol{W}_{\boldsymbol{S}}||_\infty$*. Then we have* $\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star \in \mathcal{C}$ *and the vector of coefficient estimates of BernSVM with Adaptive Lasso satisfies*

$$||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2 \leq \frac{(\frac{M}{\sqrt{n}} + ||\boldsymbol{W}_{\boldsymbol{S}}||_\infty \lambda_1)\sqrt{s}}{\kappa}. \tag{3.15}$$

The proof of Theorem 3.8 is given in Appendix **B**. In Theorem 3.8, the upper bound depends on λ_1. Therefore, a choice of λ_1 is necessary to capture the near-oracle property of the estimator Peng

et al. (2016). Note that in Theorem 3.8, we have used the fact that the first derivative of the loss function is bounded. In the next lemma, we use a probabilistic upper bound of the first derivative of the BernSVM loss function with aim to give a best choice of λ_1 and to guarantee a rate of $\mathbf{O}_P(\sqrt{s\log(p)/n})$ for the error $||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2$.

Lemma 3.9 *Let* $z_j = \frac{1}{n}\sum_{i=1}^{n} -B'_\delta(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star) y_i x_{ij}$, *and* $z^\star = \max_{\{j=1,\ldots,p\}}\{|z_j|\}$.

The variables z_j*'s are sub-Gaussian with variance* $\frac{M^2}{n}$, *and for all* $t > 0$, $z^\star$ *satisfies*

$$P(z^\star > t) \leq 2p\exp(\frac{-t^2 n}{2M^2}).$$

The proof of this lemma is given in Appendix **C**. The following theorem outlines the rate of convergence of the error $||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2$.

Theorem 3.10 *Let* $w_{max} = ||\boldsymbol{W}_S||_\infty$ *and* $w_{min} = ||\boldsymbol{W}_{S^c}^{-1}||_\infty$ *and* $\hat{\boldsymbol{\beta}}$ *the solution of the BernSVM with the adaptive Lasso. For any* $\gamma > w_{max}$, *we assume that the event* $P_1 = \{z^\star \leq \frac{(\gamma - w_{max})\lambda_1}{1+\gamma w_{min}}\}$ *is satisfied. Let* $\lambda_1 = \sqrt{2}M\frac{1+\gamma w_{min}}{(\gamma - w_{max})}\sqrt{\frac{\log(2p/\xi)}{n}}$ *for a small* ξ. *Then, under assumptions* $\boldsymbol{A}_1 - \boldsymbol{A}_3$, *we have*

$$||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2 = \boldsymbol{O}_P(\sqrt{s\log(p)/n}).$$

The proof of this theorem is similar to the proof of Theorem 3.8. It is detailed in Appendix **C**.

Remark 1 *The event* P_1, *defined in Theorem 3.10, is realized with high probability* $1 - \xi$. *This can be verified by taking* $t = \sqrt{2}M\sqrt{\frac{\log(2p/\xi)}{n}}$ *and applying Lemma 3.9, whcih leads to*

$$P(z^\star > \frac{(\gamma - w_{max})\lambda_1}{1+\gamma w_{min}}) = P(z^\star > t) \leq 2p\exp(\frac{-2M^2\log(2p/\xi)n}{2M^2 n}) = \xi.$$

In the next section we derive an upper bound for BernSVM with weighted Lasso, where the weights are estimated.

3.3.2 An upper bound of the estimation error for weighted Lasso with estimated weights

In this section, let w_j be unknown positive weights and $\hat{w}_j$ their corresponding estimates, for $j = 1, \ldots, p$, which are assumed to be non-negative. Let $\hat{\boldsymbol{\beta}}$ be a minimizer of the weighted Lasso with weights $\hat{w}_j$, defined as follows

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} B_\delta(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}) + \lambda_1 \sum_{i=1}^{p} \hat{w}_j |\beta_j|.$$

To derive an upper bound of $||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2$, we suppose that the following event, Ω_0, is satisfied Huang et Zhang (2012)

$$\Omega_0 = \{\hat{w}_j \leq w_j, \quad \forall j \in \boldsymbol{S}\} \cap \{w_j \leq \hat{w}_j, \quad \forall j \in \boldsymbol{S}^c\}.$$

The next theorem gives an upper bound of the BernSVM with weighted Lasso, where the weights are estimated.

Theorem 3.11 *Under the same conditions of Theorem 3.10 and* $\lambda_1 = \sqrt{2} M \frac{1+\gamma w_{min}}{(\gamma - w_{max})} \sqrt{\frac{\log(2p/\xi)}{n}}$, *we have*

$$||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2 = \boldsymbol{O}_P(\sqrt{s \log(p)/n}),$$

in the event $\Omega_0 \cap P_1$

The proof of this theorem is outlined in Appendix **D**.

Remark 2 *We have shown in Theorem 3.10 that the event* P_1 *is realized with high probability* $1 - \xi$. *In Theorem 3.11, we need that the event* $\Omega_0 \cap P_1$ *to be realized also with high probability. Indeed,* $P(\Omega_0 \cup P_1) = P(\Omega_0) + P(P_1) - P(\Omega_0 \cap P_1) \leq 1$ *leads to* $P(\Omega_0 \cap P_1) \geq P(\Omega_0) - \xi$.

In the next section, we extend Theorem 3.10 for non-convex penalties.

3.3.3 An extension of the upper bound for non-convex penalties

We study here the penalized BernSVM estimator with the non-convex penalties SCAD and MCP. We establish an upper bound of its error estimation under some regularity conditions.
Recall that Algorithm 3 describes steps to compute the solution of BernSVM with the non-convex

penalties, as defined in Equation (3.13). The weights $\hat{w}_j, j = 1, \ldots, p$, are computed iteratively using the first derivative of SCAD or MCP penalty. To tackle such a problem, we first search for an upper bound of $||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2$, where $\hat{\boldsymbol{\beta}}$ is an estimator of equation (3.13). The next theorem states this result.

Theorem 3.12 *Assume the same conditions of Theorem 3.10 concerning the event P_1. Let $\tilde{\boldsymbol{\beta}}$ be an initial estimator of $\boldsymbol{\beta}$ using Algorithm 5, and the weights in (3.13) are given by $\hat{w}_j = P'_{\lambda_1}(|\tilde{\beta}_j|)/\lambda_1$ for $j = 1, \ldots, p$. Then, for any $\gamma > w_{max}$ we have*

$$||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2 \leq \frac{1}{\kappa}\left(\frac{(\gamma - w_{max})\lambda_1}{1 + \gamma w_{min}} + ||P'_{\lambda_1}(|\boldsymbol{\beta}^\star_S|)||_2 + \frac{1}{a-1}||\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2\right). \tag{3.16}$$

The proof of Theorem 3.12 is given in Appendix **E**. We are now able to derive an upper bound of $||\hat{\boldsymbol{\beta}}^{(l)} - \boldsymbol{\beta}^\star||_2$, where $\hat{\boldsymbol{\beta}}^{(l)}$ is the estimator of the $l-$th iteration of Algorithm 3.

Theorem 3.13 *Assume the same conditions of Theorem 3.10 and Theorem 3.11. Let $\tilde{\boldsymbol{\beta}}$ be the minimizer of the BernSVM Lasso defined by equation (3.14), where $w_j = 1$, for $j = 1, \ldots, p$, and $\hat{\boldsymbol{\beta}}^{(l)}$ be the estimator of the $l-$th iteration of Algorithm 3. Let $\lambda_1 = \sqrt{2}M\frac{1+\gamma w_{min}}{(\gamma - w_{max})}\sqrt{\frac{\log(2p/\xi)}{n}}$. Then, we have*

$$||\hat{\boldsymbol{\beta}}^{(l)} - \boldsymbol{\beta}^\star||_2 = \boldsymbol{O}_P(\sqrt{s\log(p)/n}).$$

The proof of Theorem 3.13 is straightforward and can be outlined as follows. Since $P'_{\lambda_1}(t)$ is concave in t, we have $||P'_{\lambda_1}(|\boldsymbol{\beta}^\star_S|)||_2 \leq \sqrt{s}\lambda_1$, which means that $||P'_{\lambda_1}(|\boldsymbol{\beta}^\star_S|)||_2 = \boldsymbol{O}(\sqrt{s\log(p)/n})$. Then, by induction using equation (3.16), for each iteration, we have $||\hat{\boldsymbol{\beta}}^{(l)} - \boldsymbol{\beta}^\star||_2 = \boldsymbol{O}_P(\sqrt{s\log(p)/n})$.

3.4 Simulation and empirical studies

We study the empirical behavior of BernSVM with its competitors in terms of computational time, and we evaluate the methods estimation accuracy through different measures of performance via simulations and application to real genetic data sets.

3.4.1 Simulation study

In this section, we first compare the computation time of three algorithms : BSVM-GCD, BSVM-IRLS and HHSVM. In the second step, we examine the finite sample performance of eight regularized

SVM methods with different penalties in terms of five measures of performance.

3.4.1.1 Simulation study designs

The data sets are generated following three scenarios described bellow.

Scenario 1 : In this scenario, the columns of $\boldsymbol{X} \in \mathbb{R}^{n\times p}$, with $n = 100$ and $p = 5000$, are generated from a multivariate normal distribution with mean 0 and correlation matrix $\boldsymbol{\Sigma}$, with compound symmetry structure, where the correlation between all covariates is fixed to $\rho = 0.5$. The coefficients are set to be as follows

$$\beta_j^\star = (-1)^j \times \exp(-(2 \times j - 1)/20)$$

for $j = 1, 2, ..., 50$, and $\beta_j^\star = 0$ for $j = 51, ..., p$. Firstly, we generated

$$z_i = \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star + \epsilon_i, \quad i = 1, \ldots, n,$$

where $\epsilon_i \sim N(0, \sigma^2)$. The variance σ^2 is chosen such as a signal to noise ratio (SNR) is equal to 3, where $SNR = {\boldsymbol{\beta}^\star}^\top \boldsymbol{\Sigma}\boldsymbol{\beta}^\star/\sigma^2$. Secondly, the response variable y_i is obtained by passing z_i through an inverse-logistic function to obtain the probability $pr_i = P(y_i = 1|\boldsymbol{x}_i) = 1/(1 + exp(-z_i))$. Then, y_i is generated from a Bernouilli distribution with probability equal to pr_i. By choosing $\delta = 2, 1, 0.5, 0.1, 0.01$, we compare the computation time of the three algorithms mentioned above using lasso penalty ($\lambda_2 = 0$). The whole process is repeated 100 times and the average run time is reported in Table 1.

Scenario 2 : In this scenario, we aimed to study the impact of the correlation magnitude on the run time. As in scenario 1, we generated a data set $(\boldsymbol{x}_i, y_i), i = 1, \ldots, n$, with $n = 100$ and $\boldsymbol{x}_i \in \mathbb{R}^p$, with $p = 1000$. We consider different values for the correlation coefficient ρ. The coefficients are generated as follows $\beta_j^\star \sim U(0.9, 1.1)$ for $1 \leq j \leq 25$, $\beta_j^\star = -1$ for $51 \leq j \leq 75$ and $\beta_j^\star = 0$ otherwise. Here, $s = 50$ is the number of significant coefficients. By choosing in this scenario $\delta = 0.01, 0.5, 2$ and $\rho = 0.2, 0.5, 0.75, 0.95$, we compare the computation time of three algorithms using lasso penalty ($\lambda_2 = 0$). The whole process is repeated 100 times and The average run time is reported in Table 2.

Scenario 3 : This scenario was suggested by Christidis *et al.* (2021). Here, it is considered to compare the performance of sparse classification methods (BernSVM, sparse logistic, HHSVM

and SparseSVM) with lasso and elastic net penalties. We generated data from a logistic model

$$z_i = \log(\frac{p_i}{1-p_i}) = \beta_0 + \boldsymbol{x}_{S,i}^{\top}\boldsymbol{\beta}_S, \quad i = 1...., n,$$

where $\boldsymbol{X} \in \mathbb{R}^{n\times p}$, with $n = 50$ and $p = 800$. S is the set of active variables. The columns of $\boldsymbol{X}$ are generated from a multivariate normal with mean 0 and correlation matrix $\boldsymbol{\Sigma}$. We take $\Sigma_{ij} = \rho$ for $i \neq j$ and $\Sigma_{ii} = 1, \quad 1 \leq i, j \leq p$, which means that all the variables are equally correlated with $\rho = 0.2, 0.5, 0.8$, so the correlation matrix is different in each scenario. The number of the active predictors was set to $s = [p\xi]$, with $\xi = 0.05, 0.3$. The coefficients of the active variables are generated as $(-1)^v u$, where $v \sim Bernoulli(0.3)$ and u is uniformly distributed on $(0, 0.5)$. The response variable y_i, $i = 1, \ldots, n$, where $y_i \in \{-1, 1\}$, is generated by passing z_i through an inverse-logistic. The value of intercept β_0 is fixed such as the conditional probability $P(y_i = 1|\boldsymbol{x}_i) = 0.3$.

In this scenario, for each Monte Carlo replication, we simulated two data sets : 1) a training data of $n = 50$ observations, which is used by the different methods to perform a 10-fold cross-validation in order to choose the best model for each method ; 2) a test data of $n_{test} = 200$ observations, which is used to evaluate the methods performance. The latter is based on five measures, which are given as follows :

- The misclassification rate, MR is defined by

$$MR = \frac{\sum_{i=1}^{n_{test}}[(y_{test\,i} - \hat{y}_i)/2]^2}{n_{test}};$$

- The sensitivity SE is defined as

$$SE = 1 - \frac{\sum_{i\in A^1}[(y_{test\,i} - \hat{y}_i)/2]^2}{n_{test}},$$

  where $A^1 = \{i : y_{test\,i} = 1\}$;

- The specificity SP is defined as

$$SP = 1 - \frac{\sum_{i\in A^{-1}}[(y_{test\,i} - \hat{y}_i)/2]^2}{n_{test}},$$

  where $A^{-1} = \{i : y_{test\,i} = -1\}$;

- The precision PR is defined by

$$PR = \frac{\neq \{j : \beta_j^{\star} \neq 0, \hat{\beta}_j \neq 0\}}{\neq \{\hat{\beta}_j \neq 0\}},$$

- The recall RC is defined by

$$RC = \frac{\neq \{j : \beta_j^\star \neq 0, \hat{\beta}_j \neq 0\}}{\neq \{\beta_j^\star \neq 0\}},$$

where $\hat{y}_i$ is the $i-$th predicted for the response variable, $\beta_j^\star$ is $j-$th coordinate of the true coefficients and $\hat{\beta}_j$ is the $j-$th coordinate of the estimated coefficients. We note that the good performance of a given method in terms of SE (or SP or PR or RC) is indicated by a large value.

Of note, we have used two algorithms to solve the penalized BernSVM optimization problem : the BSVM-GCD algorithm based on an MM combined with coordinate descent and the BSVM-IRLS algorithm based on an IRLS scheme combined with coordinate descent.

The eight sparse classification methods to be compared in scenario 3 are as follows :

. BernSVM-Lasso : the BSVM-IRLS with Lasso ;

. BernSVM-EN : the BSVM-IRLS with Elastic Net ;

. Logistic-Lasso : the binomial model with Lasso computed using *glmnet* R package Friedman *et al.* (2010) ;

. Logistic-EN : the binomial model with Elastic Net computed using *glmnet* R package Friedman *et al.* (2010) ;

. HHSVM$_L$: HHSVM with Lasso computed using *gcdnet* R package Yang et Zou (2013) ;

. HHSVM$_{EN}$: HHSVM with Elastic Net computed using *gcdnet* R package Yang et Zou (2013) ;

. SparseSVM-Lasso : the sparse SVM with Lasso using the *sparseSVM* R package Yi et Huang (2017) ;

. SparseSVM-EN : the sparse SVM with Elastic Net using the *sparseSVM* R package Yi et Huang (2017).

3.4.1.2 Simulation results

δ	Time (s)		
	BSVM-GCD	BSVM-IRLS	HHSVM
0.01	7.50	**3.62**	5.84
0.1	2.09	**1.19**	1.4
0.5	1.55	**0.53**	0.86
1	1.60	**0.61**	0.77
2	2.70	**0.81**	0.85

TABLE 3.1 The run times (in seconds) for BSVM-GCD, BSVM-IRLS and HHSVM for $\delta = 2, 1, 0.5, 0.1, 0.01$ and $\lambda_2 = 0$ for Scenario 1 ($n = 100, p = 5000$).

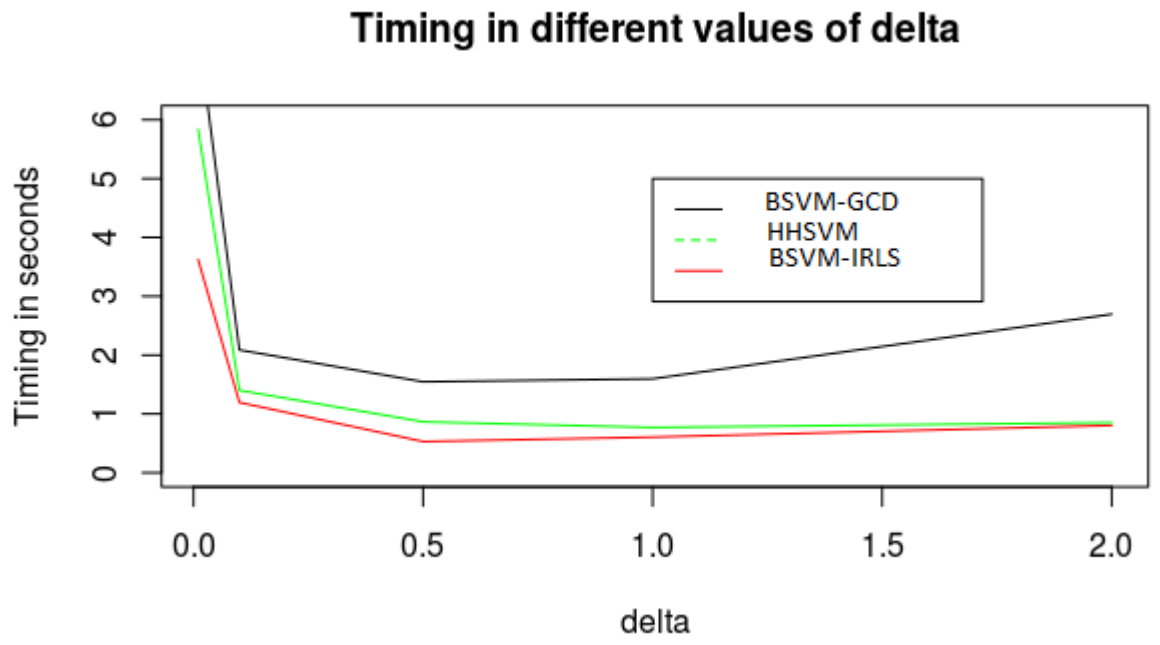

FIGURE 3.2 Comparison of the computation time of the three algorithms as a function of the parameter δ, in Scenario 1.

Summing up all the scenarios considered here, we have the following observations :

• Table 3.1 summarizes the average run time of the HHSVM and the penalized BernSVM with lasso penalty in scenario 1. Figure 3.2 provides averaged computation time curve of 100 runs of the three algorithms as a function of δ values in the same scenario. Our proposed BSVM-IRLS approach obtains the best computation time followed by HHSVM for any δ value. While BSVM-GCD produces the highest computation time for each delta value. These results are well summarized in Figure 1 by the shapes of the algorithms' computation time curves, where the curve for HHSVM is intermediate between the other two.

• Table 3.2 summarizes the average run time of the HHSVM and Bernstein penalized SVM with lasso penalty in scenario 2 with different values of δ and ρ. The result is similar to scenario 1, where BSVM-IRLS provides again the best run time followed by HHSVM for any value of δ and ρ. It is interesting to notice here that the three algorithms produce their best time for the same $\delta = 0.5$ for each value of ρ.

ρ	δ	Time (s)		
		BSVM-GCD	BSVM-IRLS	HHSVM
0.2	0.01	4.13	**2.08**	2.75
	0.5	1.06	**0.39**	0.60
	2	1.86	**0.55**	0.65
0.5	0.01	7.70	**3.52**	5.50
	0.5	1.56	**0.53**	0.90
	2	2.91	**0.85**	0.96
0.75	0.01	11.25	**4.76**	7.41
	0.5	2.31	**0.9**	1.38
	2	4.26	**1.20**	1.37
0.9	0.01	14.69	**5.84**	11.31
	0.5	2.41	**1.31**	1.37
	2	5.41	**1.77**	1.87

TABLE 3.2 Timings (in seconds) for the HHSVM and the penalized BernSVM, for $\delta = 0.01, 0.5, 2$ and $\lambda_2 = 0$, for different values of $\rho = 0.2, 0.5, 0.75, 0.9$ for Scenario 2

• Table 3.3 reports the average of the five performance measures of our methods and the competitors on Scenario 3, for $\xi = 0.05, 0.3$ and $\rho = 0.2, 0.5, 0.8$. The results obtained for scenario 3 can be summarized as follows :

- For a fixed number of active variables, increasing ρ leads to a decrease of about $5 - 14\%$ in MR, a growth of about $21 - 40\%$ in SE, and a slight increase in RC for methods using the EN penalty (except SparseSVM-EN). All methods are relatively stable in terms of SP and PR to any increase in ρ and/or in ξ.
- On the other hand, for any fixed value of ρ and for any increase in ξ, MR decreases by about $8 - 16\%$, SE decreases by about $8 - 40\%$ and RC increases slightly for only the methods based on the EN penalty and increases greatly of about $11 - 28\%$ for BernSVM-EN and HHSVM$_{EN}$.
- We observe that the methods are relatively comparable in terms of MR with a slight advantage to those using the EN penalty for any pair (ρ, ξ). While for any fixed ρ, methods based on standard penalized logistic and SVM provide good performance than the others in terms of SE for any value of ξ. The difference with methods based on the smooth approximation of the hinge loss function decreases for increasing the value of ρ and a fixed value of ξ. In addition, the BernSVM and HHSVM methods are still relatively better in terms of SP. While the methods with the lasso penalty (except SparseSVM) and logistic regression with EN penalty are better in terms of RC for any pair (ρ, ξ). Finally, the methods are relatively comparable in terms of PR.

		$\xi = 0.05$					$\xi = 0.3$				
ρ	Method	MR	SE	SP	RC	PR	MR	SE	SP	RC	PR
0.2	BernSVM-Lasso	0.32	0.14	0.96	0.04	0.06	0.22	0.39	0.95	0.04	0.33
	BernSVM-EN	0.30	0.23	0.94	0.44	0.05	0.163	0.556	0.96	0.576	0.304
	Logistic-Lasso	0.30	0.34	0.89	0.03	0.10	0.19	0.52	0.93	0.03	0.35
	Logistic-EN	0.29	0.34	0.90	0.05	0.094	0.17	0.58	0.94	0.05	0.34
	HHSVM_L	0.32	0.13	0.95	0.04	0.09	0.22	0.37	0.96	0.04	0.35
	HHSVM_{EN}	0.31	0.18	0.95	0.40	0.06	0.17	0.52	0.97	0.57	0.30
	SparseSVM-Lasso	0.29	0.34	0.89	0.19	0.06	0.16	0.61	0.94	0.25	0.30
	SparseSVM-EN	0.29	0.34	0.9	0.20	0.06	0.16	0.63	0.94	0.26	0.31
0.5	BernSVM-Lasso	0.25	0.35	0.94	0.03	0.07	0.12	0.72	0.95	0.04	0.31
	BernSVM-EN	0.23	0.38	0.96	0.43	0.05	0.08	0.8	0.96	0.54	0.30
	Logistic-Lasso	0.22	0.54	0.90	0.03	0.07	0.11	0.78	0.94	0.02	0.31
	Logistic-EN	0.21	0.55	0.91	0.04	0.07	0.09	0.81	0.95	0.05	0.32
	HHSVM_L	0.26	0.30	0.95	0.03	0.08	0.13	0.68	0.95	0.03	0.31
	HHSVM_{EN}	0.24	0.34	0.96	0.35	0.06	0.08	0.79	0.97	0.56	0.31
	SparseSVM-Lasso	0.22	0.52	0.90	0.22	0.06	0.11	0.75	0.94	0.22	0.32
	SparseSVM-EN	0.23	0.52	0.90	0.23	0.06	0.11	0.73	0.96	0.22	0.31
0.8	BernSVM-Lasso	0.19	0.54	0.95	0.02	0.06	0.08	0.81	0.97	0.03	0.31
	BernSVM-EN	0.17	0.57	0.95	0.43	0.05	0.05	0.88	0.98	0.55	0.31
	Logistic-Lasso	0.17	0.67	0.91	0.01	0.06	0.07	0.87	0.96	0.02	0.31
	Logistic-EN	0.16	0.68	0.92	0.03	0.06	0.05	0.89	0.97	0.05	0.32
	HHSVM_L	0.20	0.50	0.95	0.01	0.05	0.08	0.81	0.97	0.02	0.32
	HHSVM_{EN}	0.19	0.50	0.96	0.31	0.06	0.05	0.88	0.98	0.59	0.31
	SparseSVM-Lasso	0.19	0.60	0.91	0.30	0.06	0.11	0.78	0.94	0.33	0.32
	SparseSVM-EN	0.19	0.61	0.90	0.32	0.06	0.10	0.78	0.95	0.34	0.31

TABLE 3.3 Average of the performance measures of our methods and the competitor's on Scenario 3 for $\xi = 0.05, 0.3$ and $\rho = 0.2, 0.5, 0.8$.

Therefore, it can be concluded that our approach produced satisfactory and relatively similar results

to HHSVM.

3.4.2 Empirical study

To illustrate the effectiveness of our BernSVM approach, we also compared classification accuracy, sensitivity and specificity of the latter methods on three large-scale samples of real data sets. The first data set is the DNA methylation measurements studied in Kharoubi *et al.* (2019), and two standard data sets, usually used to evaluate classifiers performance in the literature, namely, the prostate cancer Singh *et al.* (2002) and the leukemia Golub *et al.* (1999) data sets.
The DNA methylation measurements are collected around the *BLK* gene located in chromosome 8 to detect differentially methylated regions (DMRs, refer to genomic regions with significantly different methylation levels between two groups of samples, e.g. : cases-controls). The dataset contains measurements of DNA methylation levels of 40 different samples of cell types : B cells (8 samples), T cells (19 samples), and monocytes (13 samples). Methylation levels are measured in, $p = 5896$, CpG sites (predictors). Since methylation levels are known to be different between B-cell types compared to the T- and Monocyte-cell types around the *BLK* gene Kharoubi *et al.* (2019), we coded the cell types as $y = \{-1, 1\}$ binary response, with $y = 1$ corresponds to B-cells and $y = -1$ corresponds to T- and Monocyte-cell types. The second data-set is available in a publicly R package, spls. The prostate data consists of $n = 102$ subjects, 52 patients with prostate tumor and 50 patients used as control subjects. The data contains expression of $p = 6033$ genes across the subjects. We coded the binary response variable $y = \{-1, 1\}$, with $y = 1$ corresponds to subjects with tumor prostate and $y = -1$ corresponds to normal subjects. The third data contains $n = 72$ observations and $p = 6817$ predictors and comes from a study of gene expression in two types of acute leukemias : acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML). Gene expression levels were measured using Affymetrix high density oligonucleotide arrays. Subjects with AML are coded as $y = -1$ class and subjects with ALL as $y = 1$ class. In total, we have 25 subjects in class 1 and 47 subjects in class -1. All the datasets were subjects to pre-screening procedure using a two sample t-tests Storey et Tibshirani (2003) and only 1000 predictors are retained.

For the Elastic Net regularization, we set $\lambda_2 = 3/4$. The whole process was repeated 100 times and for each data-set we trained the methods using 40% of the observations to perform a 10 cross-validation to choose regularization parameters of each method and 60% of the observations to test

Data	Prostate			Leukemia			Methylation		
Method	MR	SE	SP	MR	SE	SP	MR	SE	SP
BernSVM-Lasso	0.08	0.91	0.93	0.07	0.89	0.94	0.13	0.28	0.98
BernSVM-EN	0.08	0.9	0.94	0.04	0.95	0.96	0.08	0.68	0.97
Logistic-Lasso	0.09	0.92	0.91	0.06	0.94	0.94	0.1	0.57	0.96
Logistic-EN	0.08	0.91	0.92	0.04	0.95	0.96	0.07	0.74	0.97
HHSVM_L	0.09	0.9	0.92	0.08	0.89	0.93	0.12	0.35	0.98
HHSVM_{EN}	0.08	0.9	0.94	0.05	0.95	0.95	0.09	0.55	0.97
SparseSVM-Lasso	0.12	0.91	0.85	0.08	0.92	0.92	0.18	0.39	0.91
SparseSVM-EN	0.12	0.91	0.86	0.09	0.94	0.9	0.07	0.78	0.97

TABLE 3.4 Average of the three performance measures on three real data sets.

the methods and to compute the performance accuracy measures. The value of δ is fixed to 2 for BernSVM and HHSVM.

We see from Table 3.4 that for the prostate data, the methods with the EN penalty provide a better result in terms of MR and SP (except SparseSVM). While in terms of SE, all methods provided a comparable result with little advantage to logistic with lasso penalty. For leukemia data, methods with EN penalty (except SparseSVM) dominate for all three performance measures. For methylation data, the methods with EN penalty produce the best result in terms of MR. While SparseSVM-EN followed by Logistic-EN dominate in terms of SE. Moreover all methods are relatively comparable in terms of SP, except SparseSVM-Lasso.

3.5 Discussion

In this work we aimed to better investigate the binary classification in high dimensional setting using the SVM classifier. We proposed a new smooth loss function, called BernSVM, to approximate the SVM hinge loss function. The BernSVM loss function has nice properties, which allows for efficient implementation of penalized SVM and helps to derive appropriate theoretical results of estimators of model parameters. We have proposed two algorithms to solve the penalized BernSVM. The first one termed BSVM-GCD and combines the coordinate descent algorithm and the MM principle. The second one, called BSVM-IRLS, and uses an IRLS-type algorithm to solve the underlying optimi-

zation problem. We have also derived non asymptotic results of the penalized BernSVM estimator with weighted Lasso, with known weights, and we have extended this result for unknown weights. Furthermore, we have showed that the estimation error of penalized BernSVM with weighted Lasso achieves a rate of order $\sqrt{s\log(p)/n}$. We compared our approach with its competitors through a simulation study and the results showed that our method outperforms its competitors in terms of the computational time while maintaining good performance in terms of prediction and variable selection. The proposed work has shown also accurate results when analyzing three high-dimensional real data sets. The penalized BernSVM is implemented in an R package BSVM, which is publicly available for use from Github `https://github.com/rachidkha/BernSVM`.

Finally, given the attractive properties of the BernSVM approach to approximate penalized SVM efficiently, one can adapt the proposed method for group penalties, such as group-Lasso/SCAD/MCP. Moreover, recently an interesting penalty has been developed Christidis *et al.* (2021), which fits linear regression models that split the set of covariates into groups using a new diversity penalty. The authors provided also interesting properties of their proposed estimator. Adaptation of the diversity penalty to penalized SVM using the proposed BernSVM approach would be an interesting avenue to investigate. This is left for future work.

3.6 Appendices

3.6.1 Appendix **A**

Proof. (Theorem 3.1)

Recall the equivalent problem where

$$q_\delta(x) = g_\delta(2\delta x + 1 - \delta) \;\; x \in \;\; [0,\;\; 1],$$

must satisfy the alternative three Conditions C1', C2' and C3'. Now write

$$q_\delta(x) = \sum_{k=0}^{4} c(k,\;\; 4; q_\delta) b_{k,4}(x) \;\; , \;\; x \in \;\; [0,\;\; 1],$$

and notice that Condition C1' implies, $c(0,4;q_\delta) = \delta, c(1,4;q_\delta) = \delta/2$ *and* $c(2,4;q_\delta) = 0$. *This will be seen from the properties of the Bernstein basis above. On the other hand, Condition C2' implies that* $c(3,4;q_\delta) = 0 = c(4,4;q_\delta)$. *Therefore, we have*

$$q_\delta(x) = \delta b_{0,4}(x) + (\delta/2) b_{1,4}(x) = \delta(1-x)^4 + 2\delta x(1-x)^3, \;\; x \in \;\; [0,\;\; 1].$$

Notice that Conditions C1' and C2' uniquely determine q_δ. It remains to show that it is convex. We have

$$c(0,2;q_\delta) = 12\triangle^2 c(0,4;q_\delta) = 0,\ \ c(1,2;q_\delta) = 12\triangle^2 c(1,4;q_\delta) = 6\delta,$$

and

$$c(2,2;q_\delta) = 12\triangle^2 c(2,4;q_\delta) = 0.$$

This shows that $q_\delta^{''}(x) > 0$ for all $x \in (0,1)$ and in particular, the Condition C3' is satisfied.

In order to compute the derivatives of g_δ, we simply use the Bernstein basis properties (again) together with the chain rule as follows

$$g_\delta^{'}(t) = q_\delta^{'}((t-1+\delta)/2\delta)/2\delta,$$

and

$$g_\delta^{''}(t) = q_\delta^{''}((t-1+\delta)/2\delta)/4\delta^2,$$

with

$$q_\delta(x) = \delta b_{0,4}(x) + (\delta/2)b_{1,4}(x) = \delta(1-x)^4 + 2\delta x(1-x)^3,\ \ x \in\ [0,\ 1].$$

Thus, one has

$$q_\delta^{'}(x) = -2\delta\{b_{0,3}(x) + b_{1,3}(x)\} = -2\delta\{(1-x)^3 + 3x(1-x)^2\},$$

and

$$q_\delta^{''}(x) = 6\delta b_{1,2}(x) = 12\delta x(1-x).$$

□ *Proof. (Proposition 3.4)*

For the AEN penalty $P_{\lambda_1,\lambda_2}(\beta_j)$, the surrogate function in (3.8) can be written, for all $j = 1,\ldots,p$, as follows

$$Q_\delta(\beta_j|\beta_0,\tilde{\beta}_j) = \frac{\sum_{i=1}^n B_\delta(r_i)}{n} + \frac{\sum_{i=1}^n B_\delta^{'}(r_i)y_i x_{ij}}{n}(\beta_j - \tilde{\beta}_j) + \frac{L}{2}(\beta_j - \tilde{\beta}_j)^2 + \lambda_1 w_j|\beta_j| + \frac{\lambda_2}{2}\beta_j^2.$$

Its first derivative is given by

$$\frac{\partial Q_\delta(\beta_j|\beta_0,\tilde{\beta}_j)}{\partial\beta_j} = \frac{\sum_{i=1}^n B_\delta^{'}(r_i)y_i x_{ij}}{n} + L(\beta_j - \tilde{\beta}_j) + \lambda_1 w_j sign(\beta_j) + \lambda_2\beta_j.$$

Then, $\frac{\partial Q_\delta(\beta_j|\beta_0,\tilde{\beta}_j)}{\partial\beta_j} = 0$ implies that $\omega\beta_j = Z_j - w_j sign(\beta_j)$. Hence, Equation in (3.10) is obtained using the soft-threshold function $S(a,b)$. Equation (3.9) is a sample case of Equation (3.10) when the weights $w_j = 1$, for all $j = 1\ldots p$.

□

3.6.2 Appendix **B**

Proof. (Lemma 3.7)

Under assumption $\boldsymbol{A}2$ *on* $\boldsymbol{B}(\boldsymbol{x}_0, r_0)$*, we have*

$$\boldsymbol{H} = \frac{\boldsymbol{X}^\top \boldsymbol{D}\boldsymbol{X}}{n} \succeq \frac{\kappa_3 \boldsymbol{X}^\top \boldsymbol{X}}{n},$$

because $\boldsymbol{D} \succeq \kappa_3 \boldsymbol{I}$*, where* $\boldsymbol{M} \succeq \boldsymbol{N}$ *means that* $\boldsymbol{M} - \boldsymbol{N}$ *is positive semi-definite. The true parameter is sparse, then by Cauchy-Schwarz, we have*

$$||\boldsymbol{h}_S||_1 = \sum_{j \in \boldsymbol{S}} |h_j|.1 \leq (\sum_{j \in \boldsymbol{S}} |h_j|^2)^{1/2} (\sum_{j \in \boldsymbol{S}} 1)^{1/2} = \sqrt{s}||\boldsymbol{h}_S||_2 \leq \sqrt{s}||\boldsymbol{h}||_2.$$

We have also $\boldsymbol{h} \in \mathcal{C}$*, so we have* $||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1 \leq \gamma||\boldsymbol{h}_S||_1$*. Thus, one has*

$$\begin{aligned}
||\boldsymbol{h}||_1 &= ||\boldsymbol{h}_S||_1 + ||\boldsymbol{h}_{S^c}||_1 \\
&= ||\boldsymbol{h}_S||_1 + ||\boldsymbol{W}_{S^c}^{-1}\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1 \\
&\leq ||\boldsymbol{h}_S||_1 + ||\boldsymbol{W}_{S^c}^{-1}||_\infty ||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1 \\
&\leq ||\boldsymbol{h}_S||_1 + \gamma||\boldsymbol{W}_{S^c}^{-1}||_\infty ||\boldsymbol{h}_S||_1 \\
&= (1 + \gamma||\boldsymbol{W}_{S^c}^{-1}||_\infty)||\boldsymbol{h}_S||_1 \\
&\leq \sqrt{s}(1 + \gamma||\boldsymbol{W}_{S^c}^{-1}||_\infty)||\boldsymbol{h}||_2.
\end{aligned}$$

Then, one has

$$-\frac{\kappa_2 \log(p)}{n}||\boldsymbol{h}||_1^2 \geq -\frac{\kappa_2 s \log(p)(1 + \gamma||\boldsymbol{W}_{S^c}^{-1}||_\infty)^2}{n}||\boldsymbol{h}||_2^2.$$

Under Assumption $\boldsymbol{A}1$*, we have also*

$$\begin{aligned}
\frac{\kappa_3||\boldsymbol{X}\boldsymbol{h}||_2^2}{n} &\geq \kappa_3\kappa_1||\boldsymbol{h}||_2^2 - \kappa_3\kappa_2\frac{\log(p)}{n}||\boldsymbol{h}||_1^2 \\
&\geq \kappa_3||\boldsymbol{h}||_2^2(\kappa_1 - \frac{\kappa_2 s \log(p)(1 + \gamma||\boldsymbol{W}_{S^c}^{-1}||_\infty)^2}{n}).
\end{aligned}$$

Then, if we take

$$n > \frac{2\kappa_2 s \log(p)(1+\gamma||\boldsymbol{W}_{S^c}^{-1}||_\infty)^2}{\kappa_1},$$

we obtain

$$\kappa_1 - \frac{\kappa_2 s \log(p)(1+\gamma||\boldsymbol{W}_{S^c}^{-1}||_\infty)^2}{n} > \kappa_1 - \kappa_1/2 = \kappa_1/2.$$

Thus, we have

$$\frac{\kappa_3||\boldsymbol{X}\boldsymbol{h}||_2^2}{n} \geq \frac{\kappa_3\kappa_1}{2}||\boldsymbol{h}||_2^2,$$

which means that the Hessian matrix $\boldsymbol{H}$ satisfies the RSC with $\kappa = \frac{\kappa_3\kappa_1}{2}$. $\square$

Proof. (Theorem 3.8)

Let $\boldsymbol{h} = \hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star$. Firstly, we prove that the Lasso BernSVM error satisfies the cone constraint given by

$$\mathcal{C}(S) = \{\boldsymbol{h} \in \mathbb{R}^p : ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1 \leq \gamma||\boldsymbol{h}_S||_1\}.$$

We have, $L(\hat{\boldsymbol{\beta}}) \leq L(\boldsymbol{\beta}^\star)$ implies that $L(\boldsymbol{h}+\boldsymbol{\beta}^\star) \leq L(\boldsymbol{\beta}^\star)$. Thus,

$$\frac{1}{n}\sum_{i=1}^n B_\delta(y_i\boldsymbol{x}_i^\top(\boldsymbol{h}+\boldsymbol{\beta}^\star)) + \lambda_1||\boldsymbol{W}(\boldsymbol{h}+\boldsymbol{\beta}^\star)||_1 \leq \frac{1}{n}\sum_{i=1}^n B_\delta(y_i\boldsymbol{x}_i^\top\boldsymbol{\beta}^\star)) + \lambda_1||\boldsymbol{W}\boldsymbol{\beta}^\star||_1,$$

which implies that

$$\frac{1}{n}\sum_{i=1}^n B_\delta(y_i\boldsymbol{x}_i^\top(\boldsymbol{h}+\boldsymbol{\beta}^\star)) - \frac{1}{n}\sum_{i=1}^n B_\delta(y_i\boldsymbol{x}_i^\top\boldsymbol{\beta}^\star)) \leq \lambda_1(||\boldsymbol{W}\boldsymbol{\beta}^\star||_1 - ||\boldsymbol{W}(\boldsymbol{h}+\boldsymbol{\beta}^\star)||_1). \tag{3.17}$$

As the BernSVM loss function is twice differentiable, we can apply a second order Taylor expansion

$$\frac{1}{n}\sum_{i=1}^n B_\delta(y_i\boldsymbol{x}_i^\top(\boldsymbol{h}+\boldsymbol{\beta}^\star)) - \frac{1}{n}\sum_{i=1}^n B_\delta(y_i\boldsymbol{x}_i^\top\boldsymbol{\beta}^\star)) = \frac{1}{n}\sum_{i=1}^n B'_\delta(y_i\boldsymbol{x}_i^\top\boldsymbol{\beta}^\star)y_i\boldsymbol{x}_i^\top\boldsymbol{h} + \frac{\boldsymbol{h}^\top\boldsymbol{H}\boldsymbol{h}}{2}. \tag{3.18}$$

We can see that

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} -B_\delta^{'}(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star) y_i \boldsymbol{x}_i^\top \boldsymbol{h} &\leq \frac{1}{n}\sum_{i=1}^{n} |B_\delta^{'}(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star) y_i \boldsymbol{x}_i^\top \boldsymbol{h}| \\
&\leq \frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{x}_i^\top \boldsymbol{h}| \quad \textit{because} \quad |B_\delta^{'}(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star) y_i| = |B_\delta^{'}(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star)| \leq 1 \\
&= \frac{1}{n}\sum_{i=1}^{n} |\sum_{j=1}^{p} x_{ij} h_j| \\
&\leq \sum_{j=1}^{p} (\frac{1}{n}\sum_{i=1}^{n} |x_{ij}|) |h_j| \\
&\leq \sum_{j=1}^{p} (\frac{1}{\sqrt{n}} \sqrt{\sum_{i=1}^{n} |x_{ij}|^2}) |h_j| \\
&\leq \frac{M}{\sqrt{n}} ||\boldsymbol{h}||_1, \quad \textit{because} \quad ||\boldsymbol{x}_j||_2 \leq M, \quad \forall j.
\end{aligned}$$

Then, (4.10) and (4.11) give

$$\begin{aligned}
\frac{\boldsymbol{h}^\top \boldsymbol{H} \boldsymbol{h}}{2} &\leq \frac{1}{n}\sum_{i=1}^{n} -B_\delta^{'}(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star) y_i \boldsymbol{x}_i^\top \boldsymbol{h} + \lambda_1 (||\boldsymbol{W}\boldsymbol{\beta}^\star||_1 - ||\boldsymbol{W}(\boldsymbol{h} + \boldsymbol{\beta}^\star)||_1) \\
&\leq \frac{M}{\sqrt{n}} ||\boldsymbol{h}||_1 + \lambda_1 (||\boldsymbol{W}\boldsymbol{\beta}^\star||_1 - ||\boldsymbol{W}(\boldsymbol{h} + \boldsymbol{\beta}^\star)||_1).
\end{aligned}$$

Moreover, we have

$$\begin{aligned}
||\boldsymbol{W}\boldsymbol{\beta}^\star||_1 - ||\boldsymbol{W}(\boldsymbol{h} + \boldsymbol{\beta}^\star)||_1 &= ||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{\beta}^\star_{\boldsymbol{S}}||_1 - ||\boldsymbol{W}_{\boldsymbol{S}}(\boldsymbol{h}_{\boldsymbol{S}} + \boldsymbol{\beta}^\star_{\boldsymbol{S}})||_1 - ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1 \\
&\leq ||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1 - ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1,
\end{aligned}$$

where we have used in the first equality $\boldsymbol{\beta}^\star_{\boldsymbol{S}^c} = \boldsymbol{0}$*, and the second inequality is based on the fact that*

$$||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{\beta}^\star_{\boldsymbol{S}}||_1 - ||\boldsymbol{W}_{\boldsymbol{S}}(\boldsymbol{h}_{\boldsymbol{S}} + \boldsymbol{\beta}^\star_{\boldsymbol{S}})||_1 \leq |||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{\beta}^\star_{\boldsymbol{S}}||_1 - ||\boldsymbol{W}_{\boldsymbol{S}}(\boldsymbol{h}_{\boldsymbol{S}} + \boldsymbol{\beta}^\star_{\boldsymbol{S}})||_1| \leq ||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1.$$

Thus, we obtain

$$\begin{aligned}
0 &< \frac{\boldsymbol{h}^\top \boldsymbol{H} \boldsymbol{h}}{2} \\
&\leq \frac{M}{\sqrt{n}} ||\boldsymbol{h}||_1 + \lambda_1 (||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1 - ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1).
\end{aligned}$$

Because the right hand of inequality is positive, we have, with probability $1 - \exp(-c_0 n)$*, that*

$$\begin{aligned}
0 &\leq \frac{M}{\sqrt{n}}||\boldsymbol{h}||_1 + \lambda_1(||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1 - ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1) \\
&= \frac{M}{\sqrt{n}}||\boldsymbol{h}_{\boldsymbol{S}}||_1 + \frac{M}{\sqrt{n}}||\boldsymbol{h}_{\boldsymbol{S}^c}||_1 + \lambda_1(||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1 - ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1) \\
&= \frac{M}{\sqrt{n}}||\boldsymbol{h}_{\boldsymbol{S}}||_1 + \frac{M}{\sqrt{n}}||\boldsymbol{W}_{\boldsymbol{S}^c}^{-1}\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1 + \lambda_1(||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1 - ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1) \\
&\leq \frac{M}{\sqrt{n}}||\boldsymbol{h}_{\boldsymbol{S}}||_1 + \frac{M}{\sqrt{n}}||\boldsymbol{W}_{\boldsymbol{S}^c}^{-1}||_\infty||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1 + \lambda_1||\boldsymbol{W}_{\boldsymbol{S}}||_\infty||\boldsymbol{h}_{\boldsymbol{S}}||_1 - \lambda_1||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1 \\
&= (\frac{M}{\sqrt{n}} + \lambda_1||\boldsymbol{W}_{\boldsymbol{S}}||_\infty)||\boldsymbol{h}_{\boldsymbol{S}}||_1 - (\lambda_1 - \frac{M}{\sqrt{n}}||\boldsymbol{W}_{\boldsymbol{S}^c}^{-1}||_\infty)||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1.
\end{aligned}$$

This implies that the Lasso BernSVM error satisfies the cone constraint given by

$$\mathcal{C}(S) = \{\boldsymbol{h} \in \mathbb{R}^p : ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1 \leq \gamma||\boldsymbol{h}_S||_1\},$$

because $\lambda_1 \geq \frac{M(1+\gamma||\boldsymbol{W}_{\boldsymbol{S}^c}^{-1}||_\infty)}{\sqrt{n}(\gamma - ||\boldsymbol{W}_{\boldsymbol{S}}||_\infty)}$ *for any* $\gamma > ||\boldsymbol{W}_{\boldsymbol{S}}||_\infty$*.*

Thus, we have that the error $\boldsymbol{h}$ *belongs to the set* $\mathcal{C}(S)$*.*

Then, we derive the upper bound of the error $\boldsymbol{h}$*. From the inequality above and the definition of RSC we have*

$$\begin{aligned}
\kappa||\boldsymbol{h}||_2^2 &\leq (\frac{M}{\sqrt{n}} + \lambda_1||\boldsymbol{W}_{\boldsymbol{S}}||_\infty)||\boldsymbol{h}_S||_1 - (\lambda_1 - \frac{M}{\sqrt{n}}||\boldsymbol{W}_{\boldsymbol{S}^c}^{-1}||_\infty||)||\boldsymbol{h}_{S^c}||_1 \\
&\leq (\frac{M}{\sqrt{n}} + \lambda_1||\boldsymbol{W}_{\boldsymbol{S}}||_\infty)||\boldsymbol{h}_S||_1 \\
&\leq (\frac{M}{\sqrt{n}} + \lambda_1||\boldsymbol{W}_{\boldsymbol{S}}||_\infty)\sqrt{s}||\boldsymbol{h}||_2.
\end{aligned}$$

The second inequality comes from the fact that $(\lambda_1 - \frac{M}{\sqrt{n}}||\boldsymbol{W}_{\boldsymbol{S}^c}^{-1}||_\infty||) > 0$*. Thus, we obtain an upper bound of the error* $\boldsymbol{h} = \hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star$ *given by*

$$||\boldsymbol{h}||_2 \leq \frac{(\frac{M}{\sqrt{n}} + ||\boldsymbol{W}_{\boldsymbol{S}}||_\infty\lambda_1)\sqrt{s}}{\kappa}.$$

□ *Proof. (Theorem 3.10) In Theorem 3.8, we used* ***A2*** *and the fact that the BernSVM first derivative is bounded by 1 in order to upper bound the random variables* $z_j = \frac{1}{n}\sum_{i=1}^n -B_\delta'(y_i\boldsymbol{x}_i^\top\boldsymbol{\beta}^\star)y_i x_{ij}$ *for*

$j = 1, \ldots, p$. *Lemma 2 shows that the variables* z_j *are sub-Gaussian's. Thus, we can see that*

$$\frac{1}{n}\sum_{i=1}^{n} -B'_{\delta}(y_i \boldsymbol{x}_i^{\top}\boldsymbol{\beta}^{\star}) y_i \boldsymbol{x}_i^{\top}\boldsymbol{h} \leq z^{\star}||\boldsymbol{h}||_1,$$

where $z^{\star} = \max_j z_j$. *Following the same steps to prove Theorem 2, we obtain*

$$||\boldsymbol{h}||_2 \leq \frac{(z^{\star} + ||\boldsymbol{W}_{\boldsymbol{S}}||_{\infty}\lambda_1)\sqrt{s}}{\kappa}.$$

We have also that the event P_1 *is satisfied with high probability, then we have*

$$||\boldsymbol{h}||_2 \leq \frac{(z^{\star} + w_{\max}\lambda_1)\sqrt{s}}{\kappa} \leq \frac{\gamma(1 + w_{\min}w_{\max})\lambda_1\sqrt{s}}{\kappa}.$$

We also have that $\lambda_1 = \boldsymbol{O}(\sqrt{\log(p)/n})$. *Finally, we obtain*

$$||\boldsymbol{h}||_2 = ||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{\star}||_2 = \boldsymbol{O}_P(\sqrt{s\log(p)/n}).$$

□

3.6.3 Appendix **C**

Proof. (Lemma 3.9)

To prove Lemma 3.9, we provide some results about the sub-Gaussian random variables

Lemma 3.14 $z_1, z_2, ..., z_p$ *are* p *real random variables with* $z_j \sim subG(\sigma^2)$ *not necessarily independent, then for all* $t > 0$

$$\mathbb{P}(max_{j=1,...,p}|z_j| > t) \leq 2p\exp(-t^2/2\sigma^2).$$

Lemma 3.15 (Hoeffding's Lemma 1963) *Let* z *be a random variable such that* $\mathcal{E}[z] = 0$ *and* $z \in [a, b]$ *almost surely. Then for any* $t \in \mathbb{R}$, *it holds*

$$\mathbb{E}[e^{tz}] \leq e^{\frac{t^2(b-a)^2}{8}}.$$

In particular, $z \sim subG(\frac{(b-a)^2}{4})$.

Let $z_j = \frac{1}{n}\sum_{i=1}^{n} -B_\delta^{'}(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star) y_i x_{ij}$.

Hence, by applying Lemma 3, we conclude that z_j *is a sub-Gaussian with variance* $\frac{M^2}{n}$.

We can see that

$$\begin{aligned} |z_j| = |\frac{1}{n}\sum_{i=1}^{n} -B_\delta^{'}(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star) y_i x_{ij}| &\leq \frac{1}{n}\sum_{i=1}^{n} |B_\delta^{'}(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star)||y_i||x_{ij}| \\ &\leq \frac{1}{n}\sum_{i=1}^{n} |x_{ij}| \\ &\leq \frac{1}{\sqrt{n}}\sqrt{\sum_{i=1}^{n} x_{ij}^2} \\ &\leq \frac{M}{\sqrt{n}}. \end{aligned}$$

The third inequality by applying Cauchy-Schwartz and the fourth inequality by Assumption $\boldsymbol{A}3$.

We conclude that the variables z_j *are bounded. Moreover, using the fact that* $\boldsymbol{\beta}^\star$ *minimize the population version of the loss function and Assumption* $\boldsymbol{A}4$*, we have* $\mathbb{E}[z_j] = 0$*, then, the* z_j *are sub-Gaussian.* □

3.6.4 Appendix **D**

Proof. (Theorem 3.11)

We can see that

$$\frac{1}{n}\sum_{i=1}^{n} -B_\delta^{'}(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star) y_i \boldsymbol{x}_i^\top \boldsymbol{h} \leq z^\star ||\boldsymbol{h}||_1,$$

where $z^\star = \max_j \frac{1}{n}\sum_{i=1}^{n} -B_\delta^{'}(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star) y_i x_{ij}$.

Then, following the same argument as in the proof of Theorem 2, we have

$$
\begin{aligned}
0 &< \frac{\boldsymbol{h}^\top \boldsymbol{H}\boldsymbol{h}}{2} \\
&\leq z^\star ||\boldsymbol{h}||_1 + \lambda_1(||\hat{\boldsymbol{W}}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1 - ||\hat{\boldsymbol{W}}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1) \\
&\leq z^\star ||\boldsymbol{h}||_1 + \lambda_1(||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1 - ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1).
\end{aligned}
$$

In the fact that the right hand of inequality is positive, and the second inequality is because the event Ω_0 *is realized. Thus, we have with probability* $1 - \exp(-c_0 n)$ *that*

$$
\begin{aligned}
0 &\leq z^\star ||\boldsymbol{h}||_1 + \lambda_1(||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1 - ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1) \\
&= z^\star ||\boldsymbol{h}_{\boldsymbol{S}}||_1 + z^\star ||\boldsymbol{h}_{\boldsymbol{S}^c}||_1 + \lambda_1(||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1 - ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1) \\
&= z^\star ||\boldsymbol{h}_{\boldsymbol{S}}||_1 + z^\star ||\boldsymbol{W}_{\boldsymbol{S}^c}^{-1}\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1 + \lambda_1(||\boldsymbol{W}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1 - ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1) \\
&\leq z^\star ||\boldsymbol{h}_{\boldsymbol{S}}||_1 + z^\star w_{min}||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1 + \lambda_1 w_{max}||\boldsymbol{h}_{\boldsymbol{S}}||_1 - \lambda_1||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1) \\
&= (z^\star + \lambda_1 w_{max})||\boldsymbol{h}_{\boldsymbol{S}}||_1 - (\lambda_1 - z^\star w_{min})||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1.
\end{aligned}
$$

which implies that the Lasso BernSVM error satisfies the cone constraint given by

$$
\mathcal{C}(S) = \{\boldsymbol{h} \in \mathbb{R}^p : ||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1 \leq \gamma ||\boldsymbol{h}_S||_1\},
$$

because the event P_1 *is realized with high probability, which means that*

$$
\gamma > \frac{(z^\star + \lambda_1 w_{max})}{(\lambda_1 - z^\star w_{min})}.
$$

Thus, we have that the error $\boldsymbol{h}$ *belongs to the set* $\mathcal{C}(S)$.

Let $\alpha = \max\{w_{max}, w_{min}^{-1})\}$. *Moreover, from the inequality above and the RSC definition, we have*

$$
\begin{aligned}
\kappa||\boldsymbol{h}||_2^2 &\leq (z^\star + \lambda_1 w_{max})||\boldsymbol{h}_{\boldsymbol{S}}||_1 - (\lambda_1 - z^\star w_{min})||\boldsymbol{W}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1 \\
&\leq (w_{max} + w_{min}^{-1})\lambda_1 ||\boldsymbol{h}_S||_1 \\
&\leq 2\alpha\lambda_1\sqrt{s}||\boldsymbol{h}||_2.
\end{aligned}
$$

Then, we obtain that

$$||\boldsymbol{h}||_2 \leq \frac{2\alpha\lambda_1\sqrt{s}}{\kappa}.$$

Thus,

$$||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2 = \boldsymbol{O}_P(\sqrt{s\log(p)/n}).$$

□

3.6.5 Appendix **E**

Proof. (Theorem 3.12)

We have

$$\begin{aligned}
\kappa||\boldsymbol{h}||_2^2 &< \frac{\boldsymbol{h}^\top \boldsymbol{H}\boldsymbol{h}}{2} \\
&\leq z^\star||\boldsymbol{h}||_2 + \lambda_1(||\hat{\boldsymbol{W}}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1 - ||\hat{\boldsymbol{W}}_{\boldsymbol{S}^c}\boldsymbol{h}_{\boldsymbol{S}^c}||_1) \\
&\leq z^\star||\boldsymbol{h}||_2 + \lambda_1||\hat{\boldsymbol{W}}_{\boldsymbol{S}}\boldsymbol{h}_{\boldsymbol{S}}||_1. \star
\end{aligned}$$

For all $j \in S$, we have $\lambda_1\hat{w}_j = P'_{\lambda_1}(|\tilde{\beta}_j|)$. We have also from Taylor expansion that

$$P'_{\lambda_1}(|\tilde{\beta}_j|) = P'_{\lambda_1}(|\beta_j^\star|) + P''_{\lambda_1}(|\beta_j^\star|)(\tilde{\beta}_j - \beta_j^\star) \leq P'_{\lambda_1}(|\beta_j^\star|) + \frac{1}{a-1}(\tilde{\beta}_j - \beta_j^\star).$$

Thus from ($\star$), we obtain

$$\kappa||\boldsymbol{h}||_2^2 < ||\boldsymbol{h}||_2(z^\star + ||P'_{\lambda_1}(|\boldsymbol{\beta}_S^\star|)||_2 + \frac{1}{a-1}||\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2).$$

Therefore,

$$||\boldsymbol{h}||_2 \leq \frac{1}{\kappa}(\frac{(\gamma - w_{max})\lambda_1}{1+\gamma w_{min}} + ||P'_{\lambda_1}(|\boldsymbol{\beta}_S^\star|)||_2 + \frac{1}{a-1}||\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2).$$

□

CHAPITRE 4
THE ENSEMBLE BERNSTEIN SVM FOR CLASSIFICATION IN HIGH DIMENSION

Les méthodes ensemblistes continuent de susciter un grand intérêt chez les statisticiens en raison de leur bonne performance pour les modèles instables telles que les arbres de décision, mais ils ne sont pas faciles à interpréter. En outre, la méthode SVM régularisée produit des modèles parcimonieux et interprétables lorsqu'on considère un seul modèle. Cependant, cela peut être moins efficace pour l'analyse des données génétiques, surtout dans le cas où des gènes partagent plusieurs groupes. Dans cet article, nous proposons une nouvelle approche, appelée SplitSVM, capable d'estimer des modèles parcimonieux et faciles à interpréter. Les modèles de la méthode SplitSVM sont estimés simultanément en minimisant une fonction objective multiconvexe où la fonction de perte SVM est approchée par une fonction lisse de fonctions splines cubiques. Nous utilisons ensuite la méthode de la descente par coordonnées et la technique de la minimisation-maximisation pour résoudre le problème d'optimisation sous-jacent. La méthode SplitSVM est régularisée par deux pénalités. La première pénalité est le Lasso pour avoir des modèles parcimonieux. La deuxième pénalité est utilisée pour encourager la diversité entre les modèles de l'ensemble. L'estimateur du modèle final ou de l'ensemble est obtenu en prenant la moyenne des estimateurs des modèles formant l'ensemble. Ce modèle final est interprétable, asymptotiquement efficace, encourage la diversité entre les modèles et la parcimonie de chaque modèle. De plus, nous établissons une borne supérieure pour la norme L_2 de l'erreur estimée pour la méthode SplitSVM en présence de données de grande dimension. Les résultats des données simulées et de l'analyse de données réelles montrent les bonnes performances de la méthode SplitSVM.

Rachid Kharoubi, Abdallah Mkhadri et Karim Oualkacha

Abstract

Ensemble methods continue to attract considerable interest among statisticians because of their good performance for unstable models such as decision trees, but they are not easy to interpret. While the regularized SVMs provide sparse and interpretable models when we consider a single model, but this may be less efficient for analyzing genetic data, where genes share many several pathways. We propose a new approach, called SplitSVM, which combines the advantages of the two previous concepts. A generalized coordinate descent algorithm is proposed to solve this problem, where the combined models are estimated simultaneously by optimizing a multi-convex objective function based on a smooth SVM loss function. SplitSVM is based on two penalties. The first one is used to produce a sparse model and the second one encourages diversity among models in the ensemble. The optimal linear estimator is obtained as the average of the linear estimators of the models forming the ensemble. It is fully interpretable, asymptotically efficient and encourages diversity between models and sparsity of each model. Furthermore, we establish an L_2 bound on the non-asymptotic estimation error in high dimension for SplitSVM. The results on simulated and real data analysis show that SplitSVM is better in terms of prediction accuracy than its competitors.

Keywords: Ensemble methods ; Bernstein polynomials ; SVM ; high-dimensional binary classification ; sparsity ; diversity ; prediction accuracy

4.1 Introduction

The most commonly used standard binary classification methods in practice (medicine, economics, ...) are logistic regression (LR) and linear discriminant analysis (LDA). But these methods are less considered by the community of machines learning that prefer Support Vector Machines (SVM), Cortes et Vapnik (1995) based on the search for the optimal separating plane between the two classes. However, the goal of SVM approach is often classification accuracy, rather than statistical inference adapted to the LR and LDA framework. On the other hand, the maximum likelihood estimation of LR could be infinite if the training data are linearly separable, i. e. one can separate the two classes by a linear boundary in the feature space. While finding an optimal separating

hyperplane was the main point for SVM.

The popularity of these methods lies in their simplicity, their flexibility and their power to interpret the weights of each variable in the classification rule, especially when the dimension of the data is low or moderate. But, we notice that when the dimension increases their flexibility increases however their power of interpretation decreases further. On the other hand, the variable selection methods allow a clear improvement of the interpretation of the results at the cost of a reduction in flexibility.

Nowadays, problems with high-dimensional data, where the number of columns p in the data matrix far exceeds the number of rows n, are common in applications in health data, genetics, finance and text data. In this framework, misleading correlations between active and inactive variables are common. The regularized binary classification methods, like sparse logistic regression (Bühlmann et Van De Geer (2011), Friedman *et al.* (2010), Anzarmou et Mkhadri (2022) or sparse linear discriminant analysis Shao *et al.* (2011), Cai et Liu (2011), Mai *et al.* (2012), Fan *et al.* (2012), Anzarmou *et al.* (2023)) or sparse SVM (cf, Bradley et Mangasarian (1998), Tan *et al.* (2010), Liu *et al.* (2013) and Kharoubi *et al.* (2019)) often identify too many variables, some of which are inactive, but are reported as active. This may lead to a degradation of the prediction accuracy of the resulting classification method. All these regularized approaches, which select a subset of variables allow a clear improvement of the interpretation of the results using only one model based on one data set, but at the cost of a reduction in flexibility.

On the other hand, aggregation methods emerged during the 90s, such as SVM, under the impulse of the machine learning community. They were proposed as alternatives to tree-based segmentation methods which present several instability defects. A cleaver and simple solution they came up with consists in combining several trees (or estimators) obtained on bootstrap samples drawn according to a uniform sampling scheme or by an iterative method of weighted simulation with a weight inversely proportional to the misclassification rate. The most important approaches in this framework are Random forests (Breiman (1996),Breiman (2001)) and boosting Freund *et al.* (1996), (Friedman *et al.* (2000),Friedman (2001)) which have become very popular as "out-the- box" learning algorithms that enjoy good predictive performance.

Random forest grow many classification trees to randomized versions of the training data, and

average them. Boosting builds the trees iteratively : each tree is developed using the resulting information from the previously constructed trees without Boostrap, and hence build up an additive model consisting of a sum of trees. Their basic mechanism is based on variance or/and bias reduction. But, the interpretation of the prediction results and the selection of important variables seems more difficult for these aggregation methods. Some authors have drawn their attention to the ideas of combination and randomization of aggregation methods to stabilize the variable selection of lasso regression methods that depend heavily on the choice of the tuning parameter (cf. Bolasso by Bach (2008), Random lasso by Wang *et al.* (2010) and Randomized Lasso by Meinshausen et Bühlmann (2010). Nevertheless, the diversity between the elements of the ensemble is not taken into account to reduce their possible correlation. Recently, Christidis *et al.* (2021) proposed a novel framework for statistical learning which combines ideas from regularization and aggregation methods. They used an ensemble of logistic regression models for binary classification in high-dimensional setting. This ensemble, called Split-LR, is estimated simultaneously via optimization of a multi-objective function. They used an ℓ_1 penalty to produce sparse models and a second penalty to enforce diversity between the elements of the ensemble. Split-LR is fully interpretable as LR model, asymptotically consistent and perform better in simulation and empirical studies. However, Split-LR can inherit LR estimation problems in the case of separable data, while SVM based on optimal separating design search avoids this problem.

Motivated by Split-LR and the existence of multiple model to explain a given response McCullagh et Nelder (1989), Breiman (2001), Rudin (2019), we propose a new ensemble models, termed SplitSVM, for high-dimensional binary classification. Each element of the ensemble is a penalized smooth SVM function with lasso or the elastic net penalty for building sparse models. Instead of using randomization or other indirect method to decorrelate the models, another diversity penalty is incorporated to reduce the correlation between models, and the models are estimated jointly via optimization of a multi-objective function. So as with SplitLR, SplitSVM exploits the so-called accuracy-diversity trade-off between models which is missing in aggregation and stability selection methods. The optimal linear estimator is obtained as the average of the linear estimators of the models forming the ensemble. It is fully interpretable, asymptotically efficient and encourages diversity between models and sparsity of each model. Furthermore, we establish an L_2 bound on the non-asymptotic estimation error in high dimension for SplitSVM based on a smooth SVM loss function using Bernstein polynomials approximation Kharoubi *et al.* (2023).

This paper is organized as follows. In Section 4.2, we give more details about the construction of our ensemble method SplitSVM where each model is based on a smooth loss SVM function using the Bernstein polynomials approximation which is twice differentiable. The resulting model is a Bernstein SVM model with the power to perform the prediction accuracy and select the relevant predictors for classification performance. Then we describe, in the same section, a generalized coordinate descent algorithm to solve the solution path of the SplitSVM. In Section 4.3, we undertake the theoretical properties of the SplitSVM with lasso penalty. In particular, we establish its asymptoticall consistency and we derive a non-asymptotical upper bound of the L_2 norm of the error estimation. The empirical performance of SplitSVM based on simulation studies and application to real data sets is detailed in Section 4.4. We conclude with a brief discussion in Section 4.5.

4.2 The ensemble Bernstein SVM models (SplitSVM)

In this section, we consider the binary classification in high-dimensional setting using a data set $\mathcal{T} = (\boldsymbol{X}, \boldsymbol{y})$, where $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ is the matrix of n observations of p predictors and $\boldsymbol{y} \in \mathbb{R}^n$ with $y_i \in \{-1, 1\}$, is the binary response.

In Section 4.2.1, we first recall the penalized SVM approach. As the SVM loss function is not differentiable everywhere, we replace the hinge loss function by its recent new approximation based on the Bernstein polynomials approximation, called BernSVM Kharoubi *et al.* (2023). In Section 4.2.2, we give more details about the construction of our ensemble method, SplitSVM, where each model is based on BernSVM loss function, which is twice differentiable. The resulting model is a Bernstein SVM model with the ability to perform the prediction accuracy and select the relevant predictors in the context of the binary classification framework. An illustrative example is presented in Section 4.2.3. Then we describe, in Section 4.2.4, a generalized coordinate descent algorithm to solve the solution path of SplitSVM.

4.2.1 Bernstein SVM approach

In this section, we give a brief review of the penalized SVM framework for binary classification of high-dimensional data. A detailed discussion of SVM and its Kernel version can be found in Hastie *et al.* (2009). Then, we present a recent approximation of the SVM hinge loss function via Bernstein polynomials proposed by Kharoubi *et al.* (2023).

We assume, first, that the predictors are standardized, i.e.

$$\frac{\sum_{i=1}^{n} x_{ij}}{n} = 0 \quad \frac{\sum_{i=1}^{n} x_{ij}^2}{n} = 1, \quad 1 \leq j \leq p.$$

The SVM method Cortes et Vapnik (1995) seeks for a linear boundary hyperplane to separate the training data $\mathcal{T}$ into classes 1 and -1. The SVM classifier is the solution of the following minimization problem Hastie *et al.* (2009)

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})) + \frac{\lambda_2}{2} ||\boldsymbol{\beta}||_2^2, \tag{4.1}$$

where λ_2 is a positive tuning parameter and $\ell(t)$ is the check function defined as follows

$$\ell(t) = (1-t)_+,$$

with

$$(1-t)_+ = \begin{cases} 0, & t \geq 1 \\ 1-t, & t < 1. \end{cases}$$

The classical SVM rule in (4.1) is difficult to interpret as all predictors are used for the construction of the classification rule. To overcome this problem, recently interest has focused on an alternative class of sparse SVM approaches, which implements both variable selection and coefficient shrinkage in a single procedure. In its general form, the sparse penalized SVM problem is given by

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})) + p_{\lambda_1, \lambda_2}(\boldsymbol{\beta}), \tag{4.2}$$

where $p_{\lambda_1, \lambda_2}(\boldsymbol{\beta})$ stands for a general penalty term that may depend on positive regularization parameters λ_1 and/or λ_2. In this review section, we only consider the elastic net penalty Zou et Hastie (2003) defined as follows

$$p_{\lambda_1, \lambda_2}(\boldsymbol{\beta}) = \lambda_1 ||\boldsymbol{\beta}||_1 + \frac{\lambda_2}{2} ||\boldsymbol{\beta}||_2^2.$$

However, as the SVM hinge loss function is not differentiable everywhere, some computational difficulties can be occurred in solving problem (4.2) with respect to $\boldsymbol{\beta}$. To overcome this issue, two efficient approximations based on the Hybrid Huberized SVM loss function (HHSVM, Yang et Zou (2013)) and the Bernstein SVM (BernSVM, Kharoubi *et al.* (2023)) are recently suggested to approximate the SVM hinge loss function.

The HHSVM loss function is differentiable everywhere and its first derivative is continuous. While the BernSVM loss function is twice differentiable everywhere leading to more computational and

theoretical facilities. Kharoubi *et al.* (2023) showed the computational efficiency gain of BernSVM compared to using either HHSVM or the standard SVM hinge loss function.

The regularized BernSVM model has the objective function

$$L(\beta_0, \boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})) + P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}),$$

where $P_{\lambda_1,\lambda_2}(.)$ is the elastic net penalty, and $B_\delta(t)$ is the BernSVM loss function defined in Kharoubi *et al.* (2023) by

$$B_\delta(t) = \begin{cases} \ell(t)\ , & \text{if } |t-1| > \delta, \\ \frac{1}{8\delta^3}\{\frac{(1-t+\delta)^4}{2} - (1-t-\delta)(1-t+\delta)^3\}, & \text{if } |t-1| \leq \delta. \end{cases} \tag{4.3}$$

As mentioned before, when p largely exceeds n, misleading correlations between active and inactive variables are common in applications. The regularized binary classification methods, such as HHSVM or BernSVM, often identify too many variables, some of which are inactive, but are reported as active. This may lead to a degradation of the prediction accuracy of the resulting classification method. All these regularized approaches, which select a subset of variables, allow a clear improvement of the interpretation of the results using only one model based on one data set, but at the cost of a reduction in flexibility, which mean's that we can not explore several data sets by splitting the covariates into groups in order to learn more than one data set. In the following section, we present a new penalized SVM approach that combines ideas from regularization and aggregation to construct an ensemble of diverse and sparse models that overcome the latter problem.

4.2.2 SplitSVM models

It is clear that sparse models and aggregation methods provide a model with good generalization performance. Random forests is a good example of the judicious use of diversity between models to de-correlate the trees in order to improve the accuracy of the ensemble. However, the resulting model is difficult to interpret. Moreover, as we mentioned earlier, the ensemble methods of sparse models that are based on randomization (Random Lasso Wang *et al.* (2010), Bolasso Bach (2008) or Randomized Lasso Meinshausen et Bühlmann (2010)) do not take into account for the diversity between the components of the ensemble. The idea of combining the two concepts of sparsity and diversity in high dimension setting has recently been successfully realized in regression (Split regression Christidis *et al.* (2020) and Multi-Model Penalized Regression Wendelberger *et al.* (2020)),

and it is extended to binary classification (Split logistic regression by Christidis *et al.* (2021)). It is based on the principle of searching for, say G models (or estimators) $\hat{M}_1, \ldots, \hat{M}_G$, minimizing the sum of G sparse models with the addition of a penalty controlling the diversity between the G models.

We propose to apply this learning split principle to binary classification based on SVM classifiers. The main idea of this section is to construct an ensemble of BernSVM, which achieves good generalization performance. Then, similar to split logistic regression, we combine G linear models $(f_1, \ldots, f_G)$, where $f_g(\boldsymbol{x}_i) = \beta_0^g + \boldsymbol{x}_i^T \boldsymbol{\beta}^g$ for each observation $\boldsymbol{x}_i, i = 1, \ldots, n$, to minimize the following objective function

$$F_G((\beta_0, \boldsymbol{\beta}^1, ..., \boldsymbol{\beta}^G)) = \sum_{g=1}^{G} \left\{ \frac{1}{n} \sum_{i=1}^{n} B_\delta(y_i(\beta_0^g + \boldsymbol{x}_i^\top \boldsymbol{\beta}^g)) + P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}^g) + \frac{\lambda_d}{2} \sum_{h \neq g}^{G} P_d(\boldsymbol{\beta}^h, \boldsymbol{\beta}^g) \right\}. \quad (4.4)$$

The first penalty $P_{\lambda_1,\lambda_2}(\boldsymbol{\beta})$ is essential to prevent the over-fitting of each individual model and at the same time to induce their sparsity. We have chosen to use the elastic penalty. However, we can consider other non convex penalties such as SCAD Fan et Li (2001) or MCP Zhang (2010). While the second term is a diversity penalty ; it should favor poorly correlated models. We consider the same penalty used for split regularized regression introduced in Christidis *et al.* (2020), given as follows

$$P_d(\boldsymbol{\beta}^h, \boldsymbol{\beta}^g) = \sum_{j=1}^{p} |\beta_j^h||\beta_j^g|.$$

Therefore, models can select the same variable if only it improves their fit sufficiently. Other choices of diversity measures are possible and have been discussed in Christidis *et al.* (2021). But, this choice is preferred for its good performance on empirical data. Note that if $G = 1$, the objective function in (4.4) is reduced to the penalized BernSVM with elastic net developed in Kharoubi *et al.* (2023).

After obtaining the estimates $\hat{f}_1, \ldots, \hat{f}_G$, we take the average to form the estimate of the linear model of our ensemble method SplitSVM, given as follows

$$\hat{f}(\mathbf{x}_i) = \hat{\beta}_0 + \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}} = \frac{1}{G} \sum_{g=1}^{G} \hat{f}_g(\mathbf{x}_i),$$

where $\hat{f}_g(\mathbf{x}_i) = \hat{\beta}_0^g + \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}^g$, and $\hat{\boldsymbol{\beta}} = \frac{1}{G} \sum_{g=1}^{G} \hat{\boldsymbol{\beta}}^g$, and $\hat{\beta}_0 = \frac{1}{G} \sum_{g=1}^{G} \hat{\beta}_0^g$. Thus, SplitSVM becomes a transformation of linear function using the BernSVM loss function. This means that SplitSVM is

an BernSVM sparse model with a few variables, which we can easily interpret. Note that our goal is not only to present a powerful prediction method that performs the accuracy like the traditonnal ensemble methods but also an interpretable method that can choose the significantly variables and explain the relation between the variables and the response variable.

4.2.3 An illustrating example

In this section, we illustrate the importance of the diversity penalty in a toy example. The design matrix $\boldsymbol{X}$ was generated, as in Wendelberger *et al.* (2020), with $n = 80$ observations and $p = 6$ predictors generated from a multivariate normal distribution with an AR(1) correlation structure matrix $\Sigma(\rho)$. The correlation parameter ρ is fixed as $\rho = 0.9$ in this example. We generated a variable $\boldsymbol{z}$ using the linear model $\boldsymbol{z} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{\beta} = (1, 1, 1, 0, 0, 0)^\top$ and the entries of $\boldsymbol{\epsilon}$, ϵ_i's are independent and identically distributed (i.i.d), generated from a normal distribution with mean 0 and variance $\sigma^2 = 4$. Then, we converted the entries of $\boldsymbol{z}$ to a binary response variable, $\boldsymbol{y}$, with two classes, 1 and -1, using the inverse of the logistic function. Thus, the covariates x_1, x_2 and x_3 are associated with the response variable and the covariates x_4, x_5 and x_6 are noisy predictors.

Figure 4.1 illustrates the importance of the diversity penalty. The solution path is calculated using the SplitSVM method, with $G = 3$ models.

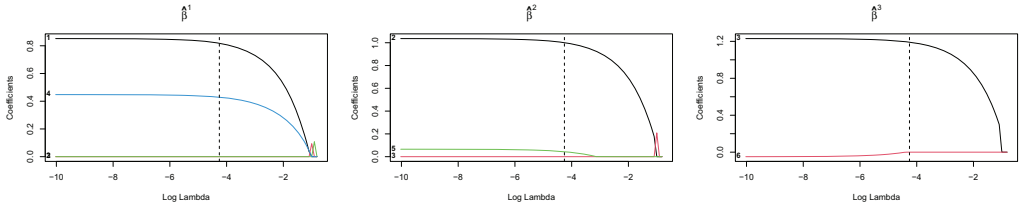

FIGURE 4.1 (a) The coefficient path of Model 1; (b) the coefficient path of Model 2; (c) the coefficient path of Model 3. The dotted vertical line in each plot shows the optimal model. The path solution is expressed as a function of a grid of the logarithm of the sparsity parameter (i.e., x-axis is expressed in the log-scale of λ_1). The parameter $\lambda_2 = 0$, and the diversity penalty parameter is fixed at its optimal value $\lambda_d = 0.378$.

We can see that Model 1 chooses the variable x_1, and in a lesser magnitude, the variable x_4. Model 2 selects the variable x_2, and then the variable x_5 in a lesser magnitude. Model 3 captures the x_3 variable. Thus, aggregating these models in SplitSVM will result on selection of x_1, x_2 and x_3.

In the next section, we describe a computationally-efficient algorithm to solve SplitSVM.

4.2.4 Algorithm for SplitSVM

Let $\Phi = (\beta_0^1, \boldsymbol{\beta}^1, \ldots, \beta_0^G, \boldsymbol{\beta}^G)$ be the vector of unknown $G(p+1)$ parameters, which can be partitioned in G classes, $C_g = \{(\beta_0^g, \boldsymbol{\beta}^g)\}, g = 1, \ldots, G$, of dimension $(p+1)$ each, that is, $\Phi = \cup_{g=1}^{G} C_g$. The presence of the non-convex diversity penalty leads to the difficulty of obtaining a global minimum of problem (4.4). To circumvent this difficulty, we observe that the objective function is convex in each class C_g while keeping the other $(G-1)(p+1)$ parameters fixed at the current estimates $(\tilde{\beta}_0^h, \tilde{\boldsymbol{\beta}}^h)$ for $h \neq g$. So, the objective function to be minimized on each class C_g becomes

$$F(\beta_0^g, \boldsymbol{\beta}^g) = \frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i(\beta_0^g + \boldsymbol{x}_i^\top \boldsymbol{\beta}^g)) + P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}^g) + \frac{\lambda_d}{2}\sum_{h\neq g}^{G} P_d(\tilde{\boldsymbol{\beta}}^h, \boldsymbol{\beta}^g). \tag{4.5}$$

The latter problem (4.5) can be written as a penalized BernSVM problem with an adaptive elastic penalty as follows

$$F(\beta_0^g, \boldsymbol{\beta}^g) = \frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i(\beta_0^g + \boldsymbol{x}_i^\top \boldsymbol{\beta}^g)) + \frac{\lambda_2}{2}||\boldsymbol{\beta}^g||_2^2 + \sum_{j=1}^{p} \tilde{w}_j^g |\beta_j^g|, \tag{4.6}$$

where $\tilde{w}_j^g = \lambda_1 + \frac{\lambda_d}{2}\sum_{h\neq g} |\tilde{\beta}_j^h|$. The minimizer of the objective function (4.6) is given in the next proposition.

Proposition 4.1 *For fixed λ_1, λ_2 and λ_d, the problem (4.6) can be solved using coordinate descent algorithm and the coordinate wise update for β_j^g is given by*

$$\hat{\beta}_j^g = \frac{S(Z_j^g, \tilde{w}_j^g)}{\frac{3}{4\delta} + \lambda_2}, \tag{4.7}$$

where $Z_j^g = -\frac{\sum_{i=1}^{n} B'_\delta(m_i^g) y_i x_{ij}}{n} + \frac{3}{4\delta}\tilde{\beta}_j^g$, $m_i^g = y_i(\tilde{\beta}_0^g + \boldsymbol{x}_i^\top \tilde{\boldsymbol{\beta}}^g)$, *and*
$S(a,b) = \max(|a| - b, 0) sign(a)$.

The proof of this proposition is similar to the proof of Proposition 3.4 in chapter 3 Kharoubi *et al.* (2023). Thus, it is omitted here.

For λ_2, λ_d fixed and a grid of λ_1, to solve the objective function in (4.5), we use the following Algorithm 6, which is based on coordinate descent and MM principle.

Algorithm 6 The SplitSVM algorithm to solve (4.5) with elastic net penalty.

— Initialize $\tilde{\beta}_0$ and $\tilde{\boldsymbol{\beta}}$,

— Cyclic coordinate descent for $g = 1, ..., G$

1. Compute $\hat{\beta}_0^g$ and $\hat{\boldsymbol{\beta}}^g$ using the following steps (a) and (b)
   (a) First, compute $m_i^g = y_i(\tilde{\beta}_0 + \boldsymbol{x}_i^\top \tilde{\boldsymbol{\beta}})$, then compute $\hat{\beta}_j^g$ using (4.4) and set $\tilde{\beta}_j = \hat{\beta}_j^g$;
   (b) Recompute $m_i^g = y_i(\tilde{\beta}_0 + \boldsymbol{x}_i^\top \tilde{\beta})$, then compute $\hat{\beta}_0^g = \tilde{\beta}_0 - L^{-1} \frac{\sum_{i=1}^n B'_\delta(m_i^g) y_i}{n}$;
2. Set $\tilde{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}^g$ and $\tilde{\beta}_0 = \hat{\beta}_0^g$.

After each cyclic update in Algorithm 6, we declare convergence if

$$\max_{0 \le j \le p} \left(\frac{1}{G} \sum_{g=1}^{G} \hat{\beta}_j^g - \frac{1}{G} \sum_{g=1}^{G} \tilde{\beta}_j^g \right)^2 < e^{-5}.$$

4.2.4.1 Choosing the tuning parameters λ_1 and λ_d

For each λ_2 fixed, we compute the solutions of the final output using a grid of λ_1 and λ_d^{opt}, where λ_d^{opt} is the optimal value of λ_d corresponding to the minimal value of a CV criterion (more details are given below). We describe in what follows how to obtain the grid of λ_1 and λ_d^{opt}. Firstly, we start by constructing the grid of λ_1 in the special case where $\lambda_d = 0$ by computing the $\lambda_{1,max}^0$ that makes all models null. In fact, $\lambda_{1,max}^0 = \frac{1}{n} \max_j |\sum_{i=1}^n B'_\delta(\hat{m}) y_i x_{ij}|$, where $\hat{m} = arg\min_m \frac{1}{n} \sum_{i=1}^n B_\delta(y_i m)$. For $\lambda_d > 0$, the KKT conditions for problem (4.6) are

$$-\frac{\sum_{i=1}^n B'_\delta(m_i^g) y_i x_{ij}}{n} = (\frac{\lambda_d}{2} \sum_{h \neq g} |\tilde{\beta}_j^h| + \lambda_1) sign(\beta_j^g) \quad \text{if} \quad \beta_j^g \neq 0,$$

and

$$|\frac{\sum_{i=1}^n B'_\delta(m_i^g) y_i x_{ij}}{n}| \le \lambda_1 + \frac{\lambda_d}{2} \sum_{h \neq g} |\tilde{\beta}_j^h| \quad \text{if} \quad \beta_j^g = 0,$$

for $j = 1, \ldots, p$. As a result of the KKT conditions, one can see that $\lambda_{1,\max}$ needed to force all β_j's to be non-active (i.e. to be zero) is less than $\lambda_1^{max,0}$ because $(\lambda_d/2) \sum_{h \neq g} |\tilde{\beta}_j^h| > 0$ for $\lambda_d > 0$. Thus, $\lambda_{1,\max} \le \lambda_{1,max}^0$. Although a grid search might be feasible to numerically obtain $\lambda_1^{\max}$, empirical studies showed that setting $\lambda_{1,\max} = \lambda_{1,max}^0$ yields good results for both $\lambda_d = 0$ and $\lambda_d \neq 0$. In our algorithm implementation, we take $\lambda_{1,\max} = \lambda_{1,max}^0$, for all values of λ_d. Thus, the grid of λ_1 is

determined as follows : we set $\lambda_1^{min} = \tau\lambda_{1,max}$, where $\tau = 10^{-2}$ for $n < p$ and $\tau = 10^{-4}$ for $n \geq p$. The grid is obtained by placing, in the log-scale, uniformly 98 points between $\lambda_{1,min}$ and $\lambda_{1,max}$. Now, for a fixed value of λ_1, to compute λ_d^{opt} one needs first to determine $\lambda_{d,max}$, which is the smallest value of λ_d that makes the models disjoint, then one can build a grid of values of λ_d between 0 and $\lambda_{d,max}$. Finally, cross-validation can be used to chose the optimal value λ_d^{opt}.

The search of the SplitSVM optimal model is described in the steps below :

— start by taking $\lambda_d^{opt} = 0$;

— for the grid of λ_1, compute the solution path of SplitSVM as described in Algorithm 6, and use 10-cross-validation to obtain λ_1^{opt} ;

— fix $\lambda_1 = \lambda_1^{opt}$, set λ_d to a large value (e.g., $\lambda_d = G$), and compute the grid of λ_d in a similar way as the grid of λ_1 was calculated above. Set $\lambda_{d,\max}$ to be the smallest λ_d that makes the models disjoint ;

— calculate a new grid of λ_d between $\lambda_d = 0$ and $\lambda_{d,\max}$;

— do another 10-cross-validation to obtain λ_d^{opt} ;

— repeat these processes many times until no further improvement in the value of the CV criterion with respect to optimal λ_d^{opt} ;

— fix λ_d at λ_d^{opt}, and compute the final path solution of SplitSVM with respect to λ_1.

Of note, the CV criterion used to determine optimal values of both λ_1 and λ_d is the error margin based on the SplitSVM loss function on a test set ; it is defined as

$$CV_{err} = \frac{2}{n_{test}} \sum_{i=1}^{n_{test}} B_\delta(y_i(\hat{\beta}_0 + \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}})). \tag{4.8}$$

SplitSVM is implemented in an R package called SplitSVM available at `https://github.com/rachidkha/SplitSVM`.

4.2.4.2 Illustration of the role of diversity penalty

In order to more understanding the role of the diversity penalty, we study the case of two models ($G = 2$) with a design matrix $\boldsymbol{X} \in \mathbb{R}^{n\times 2}$ with two correlated predictors $\boldsymbol{x}^j \in \mathbb{R}^n$ with $j = 1, 2$. Without loss of generality, we suppose that $\beta_0^j = 0$ for $j = 1, 2$. The following proposition provides the explicit estimates of the two coefficients, under two scenarios, during a current iteration of Algorithm 1.

Proposition 4.2 *We assume that $\hat{\boldsymbol{\beta}}$ is a Split BernSVM solution. Define the margin m_i^g or $m_i^h = y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^h$ with $g = 1, 2 \neq h = 1, 2$. Let $\tilde{y}_i = y_i B_\delta'(m_i^g$ or $m_i^h)$, for $i = 1, ..., n$, and $r_j = \tilde{\boldsymbol{y}}^\top \boldsymbol{x}^j/n$ for $j = 1, 2$.*

1. *If Model 1 and Model 2 are disjoints, then the active coefficients take the value*

$$\hat{T}_j = \frac{S(r_j, \lambda_1)}{\frac{3}{4\delta} + \lambda_2}, \quad j = 1, 2,$$

and

$$\lambda_d/2 \geq \max\left\{ \frac{|-r_1 + \frac{3}{4\delta}\hat{T}_1| - \lambda_1}{|\hat{T}_2|}, \quad \frac{|-r_2 + \frac{3}{4\delta}\hat{T}_2| - \lambda_1}{|\hat{T}_1|} \right\}.$$

2. *Assume that $\hat{\beta}_1^1 = \hat{\beta}_1^2 = 0$, which means that the first variable is inactive in Model 1 and Model 2. Assume that the second variable is active in both models, then their coefficients are given as follows*

$$\hat{\beta}_2^1 = \frac{S(-r_2 + \frac{3}{4\delta}\hat{\beta}_2^2, \lambda_1 + \frac{\lambda_d}{2}|\hat{\beta}_2^2|)}{\frac{3}{4\delta} + \lambda_2},$$

and

$$\hat{\beta}_2^2 = \frac{S(-r_2 + \frac{3}{4\delta}\hat{\beta}_2^1, \lambda_1 + \frac{\lambda_d}{2}|\hat{\beta}_2^1|)}{\frac{3}{4\delta} + \lambda_2}.$$

Proof. In the first case, we have Model 1 and Model 2 are disjoints. Therefore, we suppose that $\hat{\beta}_1^2 = \hat{\beta}_2^1 = 0$. Then we obtain $\tilde{w}_1^1 = \tilde{w}_2^2 = \lambda_1$. From Equation (4.7) we obtain

$$\hat{\beta}_1^1 = \hat{T}_1, \quad \hat{\beta}_2^2 = \hat{T}_2.$$

We have also

$$\hat{\beta}_1^2 = \frac{S(-\frac{\sum_{i=1}^n B_\delta'(m_i^2) y_i x_i^1}{n} + \frac{3}{4\delta}\hat{T}_1, \lambda_1 + \frac{\lambda_d}{2}|\hat{T}_1|)}{\frac{3}{4\delta} + \lambda_2},$$

and

$$\hat{\beta}_2^1 = \frac{S(-\frac{\sum_{i=1}^n B_\delta'(m_i^1) y_i x_i^2}{n} + \frac{3}{4\delta}\hat{T}_2, \lambda_1 + \frac{\lambda_d}{2}|\hat{T}_2|)}{\frac{3}{4\delta} + \lambda_2}.$$

It must be

$$|-\frac{\sum_{i=1}^n B_\delta'(m_i^1) y_i x_i^2}{n} + \frac{3}{4\delta}\hat{T}_2| \leq \lambda_1 + \frac{\lambda_d}{2}|\hat{T}_2|, \quad |-\frac{\sum_{i=1}^n B_\delta'(m_i^2) y_i x_i^1}{n} + \frac{3}{4\delta}\hat{T}_1| \leq \lambda_1 + \frac{\lambda_d}{2}|\hat{T}_1|.$$

Thus we obtain $\lambda_d/2 \geq \max\left\{ \frac{|-r_1 + \frac{3}{4\delta}\hat{T}_1| - \lambda_1}{|\hat{T}_2|}, \quad \frac{|-r_2 + \frac{3}{4\delta}\hat{T}_2| - \lambda_1}{|\hat{T}_1|} \right\}$. □

To interpret this proposition, we take $\lambda_1 = o(r_1) = o(r_2)$. In this case, if Model 1 and Model 2 are disjoints we obtain

$$\lambda_d/2 \geq \max\left\{\frac{|-r_1+\frac{3}{4\delta}\hat{T}_1|}{|\hat{T}_2|}, \quad \frac{|-r_2+\frac{3}{4\delta}\hat{T}_2|}{|\hat{T}_1|}\right\}.$$

This leads to

$$\lambda_d/2 \geq \max\left\{\frac{|(1-\frac{3}{4\delta})r_1|}{|r_2|}, \quad |\frac{(1-\frac{3}{4\delta})r_2|}{|r_1|}\right\}.$$

If the two predictors are associated with the variable response with same magnitude, which means that $|r_1| = |r_2|$. This occurs when

$$|\sum_{i=1}^{n} B'_\delta(m_i^g \ \ or \ \ m_i^h)x_i^1| = |\sum_{i=1}^{n} B'_\delta(m_i^g \ \ or \ \ m_i^h)x_i^2|.$$

The diversity parameter required to separate the two models is $\lambda_d/2 = |1-\frac{3}{4\delta}|$. Since $|1-\frac{3}{4\delta}|$ is fixed, the diversity parameter depends only on the association of the predictors with the response variable.

4.3 Theoretical analysis

In this section, we establish that SplitSVM is asymptotically consistent in the moderate setting when $\log(p)/n$ goes to zero. Moreover, we derive an upper bound of the L_2 norm of the error estimation $||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2$, where $\boldsymbol{\beta}^\star$ is the minimizer of the population version without the penalty term ; i.e., $\boldsymbol{\beta}^\star = argmin_{\boldsymbol{\beta}}\mathbb{E}[B_\delta(y\boldsymbol{x}^\top\boldsymbol{\beta})]$.

Let $S = \{j : \beta_j^\star \neq 0\}$ be the set of active predictors, with $s = |S|$ the number of elements of S, and let S^c be complement S. For seek of simplicity, the intercept is omitted for the theoretical development.

4.3.1 A non-asymptotically upper bound of the SplitSVM error estimation

At first, we derive an upper bound of the L_2 norm of the error $\boldsymbol{h} = \hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star$ for each $g = 1, \ldots, G$. Then, we deduce an upper bound of the L_2 norm of the difference $\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star$ where $\hat{\boldsymbol{\beta}} = \frac{1}{G}\sum_{g=1}^{G}\hat{\boldsymbol{\beta}}^g$. Without loss of generality, we assume that $\lambda_2 = 0$. We state the same assumptions as in Kharoubi *et al.* (2023) :

— **A1 :** Define the convex cone

$$\mathcal{C}(S) = \{\boldsymbol{h} \in \mathbb{R}^p : ||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1 \leq \gamma||\boldsymbol{h}_S||_1\},$$

where $\boldsymbol{W}$ is a diagonal matrix with unknown weights and $\gamma > 0$. The matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ satisfies, for all $\boldsymbol{h} \in \mathcal{C}$, the condition

$$\frac{||\boldsymbol{X}\boldsymbol{h}||_2^2}{n} \geq \kappa_1 ||\boldsymbol{h}||_2^2 - \kappa_2 \frac{\log(p)}{n} ||\boldsymbol{h}||_1^2,$$

with probability $1 - \exp(-c_0 n)$, for some $c_0 > 0$, where the rows of $\boldsymbol{X}$, $\boldsymbol{x}_i$, are i.i.d Gaussian or Sub-Gaussian, and κ_1 and κ_2 are two positive constants.

— **A2 :** There exists $\boldsymbol{B}(\boldsymbol{x}_0, r_0)$ a ball centered at $\boldsymbol{x}_0$ with radius r_0, such as for all $\boldsymbol{x} \in \boldsymbol{B}(\boldsymbol{x}_0, r_0)$, we have $B_\delta^{''} \circ h^\star(\boldsymbol{x}) > \kappa_3 > 0$, where $h^\star(\boldsymbol{x}) = \boldsymbol{y}\boldsymbol{x}\boldsymbol{\beta}^\star$.

— **A3 :** The columns of the design matrix $\boldsymbol{X}$ are bounded :

$$\text{for all} \quad j \in \{1, \ldots, p\} \quad ||\boldsymbol{x}_j||_2 \leq M.$$

Assumption **A1** is satisfied with high probability for Gaussian random matrices Raskutti *et al.* (2010). Rudelson et Zhou (2012) extended this result to the Sub-Gaussian random matrices. The existence of $\boldsymbol{x}_0$ in Assumption **A2** is guaranteed by the behavior of the second derivative $B_\delta^{''}(.)$. Indeed, for $\delta > 0$, we have $B_\delta^{''}(1) = \frac{3}{4\delta} > 0$. We can take $\boldsymbol{x}_0 \in (h)^{\star -1}(\{1\}) \subset \mathbb{R}^p$, where $(h)^\star(\boldsymbol{x}_0) = y\boldsymbol{x}_0^\top \boldsymbol{\beta}^\star$. As the second derivative of the BernSVM loss function exists and continuous, the function $B_\delta^{''} \circ (h)^\star(\boldsymbol{x}_0)$ is also continuous in $\boldsymbol{x}_0$ and satisfies $B_\delta^{''} \circ (h)^\star(\boldsymbol{x}_0) > 0$. The existence of a ball around $\boldsymbol{x}_0$ as described in **A2** is a result of the continuity properties of $B_\delta^{''} \circ (h)^\star(.)$.
Assumption **A2** is needed to ensure the positive-definiteness of the Hessian matrix around $\boldsymbol{\beta}^\star$ in order to derive an upper bound of the L_2 norm of $||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||$. See Lemma 1 in Appendix **A** for more detail concerning the use of this assumption. Note that in the case of studying theoretical properties of the standard SVM model, to reach a similar property for the Hessian, Koo *et al.* (2008) assumed three assumptions, namely **A1**, **A3** and **A4**; see Lemma 5 of Koo *et al.* (2008).

For each model, $g = 1, \ldots, G$, we have

$$\hat{\boldsymbol{\beta}}^g = \arg\min_{\boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^n B_\delta(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^g) + \lambda_1 ||\boldsymbol{\beta}^g||_1 + \frac{\lambda_d}{2} \sum_{h \neq g}^G \sum_{j=1}^p |\beta_j^h| |\beta_j^g|.$$

We consider the objective function given as follows

$$L(\boldsymbol{\beta}^g) = \frac{1}{n} \sum_{i=1}^n B_\delta(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^g) + \lambda_1 ||\boldsymbol{\beta}^g||_1 + \frac{\lambda_d}{2} ||\boldsymbol{W}\boldsymbol{\beta}^g||_1,$$

where $\boldsymbol{W}$ is a diagonal matrix with unknown weights $w_j \geq 0$, for $j = 1, \ldots, p$. Let $\hat{\boldsymbol{W}}$ an estimate of $\boldsymbol{W}$ with $\hat{w}_j = \sum_{h \neq g}^G |\hat{\beta}_j^h|$.

As in Huang and Zhang (2012), we define the event

$$\Omega_0 = \{\hat{w}_j \leq w_j, \forall j \in S\} \cup \{w_j \leq \hat{w}_j, \forall j \in S^c\}.$$

The following theorem gives an upper bound of the L_2 estimation error $||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2$.

Theorem 4.3 *Assume that Assumptions* $\boldsymbol{A}1-\boldsymbol{A}3$ *are met and* $\lambda_d/2 > (\frac{M}{\sqrt{n}}+\lambda_1)||\boldsymbol{W}_S{}^{c-1}||_\infty$. *Then, we have for each* $g = 1, ..., G$ *that* $\hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star \in \mathcal{C}(S)$ *and we have also that the vector of SplitSVM coefficient estimates satisfies*

$$||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2 \leq \frac{(\frac{M}{\sqrt{n}} + \lambda_1 + \frac{\lambda_d}{2}||\boldsymbol{W}_S||_\infty)\sqrt{s}}{\kappa}, \tag{4.9}$$

where $\kappa > 0$. *If we choose* λ_1 *and* λ_d *to be of order* $\sqrt{\log(p)/n}$, *we obtain an upper bound in 4.9 of order* $\sqrt{s\log(p)/n}$.

The proof of this theorem is deferred to Appendix **A**.

4.3.2 Asymptotically consistent property for SplitSVM

In this section, we derive a consistent property of the prediction error of SplitSVM. Let define the populational risk as follows

$$R(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[B_\delta(y_i f(\boldsymbol{x}_i))],$$

where $f(.)$ is any p-dimensional function belongs to the function space $\mathcal{H}$ which includes $\mathcal{H}_\ell$, the linear transformations subspace. Let $f^\star = arg\min_{f\in\mathcal{H}} R(f)$ not necessarily linear, and $f_{\boldsymbol{\beta}^\star}$ is the best linear approximation of $f^\star$, where $f_{\boldsymbol{\beta}^\star} = arg\min_{f\in\mathcal{H}_\ell} R(f)$. The excess error is defined as follows

$$\mathcal{E}(f) = R(f) - R(f^\star).$$

Let also $\Lambda_{\max}$ be the largest eigenvalue of $\boldsymbol{X}^\top\boldsymbol{X}/n$. The next Theorem shows that prediction error can be consistent under some specific conditions.

Theorem 4.4 *Let* $\hat{\boldsymbol{\beta}}$ *be the solution of the SplitSVM problem and* $\boldsymbol{\beta}^\star$ *is the true coefficient vector. Then, under the same conditions of Theorem 4.3, we obtain*

$$\mathcal{E}\left(\frac{1}{G}\sum_{g=1}^{G} f_{\hat{\boldsymbol{\beta}}^g}\right) \leq 2\mathcal{E}(f_{\boldsymbol{\beta}^\star}) + \frac{L}{2G}\Lambda_{\max} n \sum_{g=1}^{G} ||\hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star||_2^2.$$

If we have $f^\star = f_{\boldsymbol{\beta}^\star}$ *and* $\Lambda_{\max} = \boldsymbol{o}(\frac{1}{s \log p})$*, then* $\frac{1}{G}\sum_{g=1}^{G} f_{\hat{\boldsymbol{\beta}}^g}$ *is consistent.*

The proof of this theorem is deferred to Appendix **B**.

4.4 Simulation study and real data-sets analysis

In this section, we examine the performance of SplitSVM based on three simulation scenarios, similar to those explored in Christidis *et al.* (2021). In all scenarios, the performance of SplitSVM is compared to existing methods ; more detail concerning the competitors is given in Section 4.4.2. All the studied scenarios are implemented in the *R* software.

4.4.1 Scenarios of simulation

For each scenario, we simulate two data sets : 1) a training data of $n = 50$ observations, which is used by all the methods to perform a 10-fold cross-validation with aim to choose the best model for each approach ; 2) a test data of 200 observations to evaluate the methods' performance by computing different measures described below. The number of predictors is fixed to $p = 800$. We generate the data from a logistic model :

$$z_i = \log(\frac{p_i}{1-p_i}) = \beta_0 + \boldsymbol{x}_{S,i}^\top \boldsymbol{\beta}_S, \quad i = 1...., n,$$

where $\boldsymbol{x}_{S,i}$ is the set of the active covariates for subject i, $\boldsymbol{\beta}_S$ is the vector of effects of the active covariates, and S is the set of active covariates. The size of the active set is $|S| = s = [p\xi]$, where $\xi = 0.05, 0.3$. The coefficients of the active variables are generated as $(-1)^v u$, where $v \sim Bernoulli(0.3)$ and u is uniformly distributed on $(0, 0.5)$.

The response variable, $y_i \in \{-1, 1\}$, is generated from the Bernoulli distribution, with probability of success p_i, by passing z_i through the inverse-logistic function ; i.e., $p_i = logit^{-1}(z_i)$. The value of β_0 is fixed such as the marginal probability $P(y_i = 1) = 0.3$.

The difference between the three scenarios is based on the dependence structure between the covariates. That is, the design matrix of the predictors is simulated using three different correlation structures. More precisely, the columns of the design matirx, $\boldsymbol{X}$, are generated from a multivariate normal distribution, with mean $\mathbf{0}_p$ and correlation matrix $\boldsymbol{\Sigma}$ as follows :

— In **Scenario** 1, we take $\Sigma_{ij} = \rho$, for $i \neq j$, $1 \leq i \leq p$, which means that all the predictors are equally correlated.

— In **Scenario 2**, for $i \neq j$, we set $\Sigma_{ij} = \rho$ for each two (i, j) active variables, and $\Sigma_{ij} = 0$ otherwise.

— In **Scenario 3**, we assume that the correlation matrix has a block-structure. More precisely, the active predictors are split to $B = \xi p/20$ disjoints blocks, with each block contains 20 predictors, and we set $\Sigma_{ij} = \rho$ for the couples (i, j) of active variables in each block, and $\Sigma_{ij} = 0.2$ for couples belonging to different blocks. Finally, the correlation between the active and non-active variables is also set to be 0.2.

For all scenarios, we varied the value of $\rho \in \{0.2, 0.5, 0.8\}$.

4.4.2 Methods

The six sparse classification methods to be compared are as follows :

1. SplitSVM with Lasso, called SplitSVM-Lasso.
2. SplitSVM with Elastic Net, called SplitSVM-EN.
3. Logistic regression with either Lasso or EN, implemented in *glmnet* R package.
4. The ensemble Logistic Regression with Lasso and Elastic Net, called SplitLR-Lasso and SplitLR-EN ; both methods are implemented in *SplitGLM* R package.

For the ensemble methods, SplitSVM and SplitLR, the number of models is fixed to $G = 5$, according to an optimal criterion for choosing the number of diverse groups, G. This is described in more details in Section 4.4.5. The choice of the best (λ_1, λ_d) is explained in Section 4.2.4.1.

4.4.3 Performance measures

The simulation study aims to evaluate the performance of SplitSVM and its competitors in terms of the prediction accuracy using five measures of performance defined as follows :

- Misclassification rate, MR is defined by

$$MR = \frac{\sum_{i=1}^{n_{test}} [(y_{test_i} - \hat{y}_i)/2]^2}{n_{test}};$$

- Sensitivity SE is defined as

$$SE = 1 - \frac{\sum_{i \in A^1} [(y_{test_i} - \hat{y}_i)/2]^2}{n},$$

where $A^1 = \{i : y_{test_i} = 1\}$;

- Specificity SP is defined as

$$SP = 1 - \frac{\sum_{i \in A^{-1}} [(y_{test_i} - \hat{y}_i)/2]^2}{n},$$

where $A^{-1} = \{i : y_{test_i} = -1\}$;

- Precision PR is defined by

$$PR = \frac{\neq \{j : \beta^\star{}_j \neq 0, \hat{\beta}_j \neq 0\}}{\neq \{\hat{\beta}_j \neq 0\}};$$

- Recall RC is defined by

$$RC = \frac{\neq \{j : \beta^\star{}_j \neq 0, \hat{\beta}_j \neq 0\}}{\neq \{\beta^\star{}_j \neq 0\}},$$

where $\hat{y}_i$ is the predicted response, $\beta^\star_j$ is the $j-th$ coordinate of the true coefficients and $\hat{\beta}_j$ is $j-th$ estimated coefficient. We note that the good performance of a given method is indicated by a largest value of the statistics SE, SP, PR, RC. For the MR statistic, good performance corresponds to a small value of MR.

4.4.4 Results of Simulation studies

In this section, we report the results of the simulation Scenario 1, Scenario 2 and Scenario 3. Tables 4.1, 4.2 and 4.3 summarize the results of each scenario according the parameter-combination of ξ and ρ.

From Table 4.1, one can summarizes the results of Scenario 1, for the six competitor methods, as follows :

— In terms of MR, the ensemble methods provide the best results for any configuration of (ρ, ξ), with a slight advantage for SplitLR.

— SplitSVM is very efficient in terms of SE whatever the configuration of (ρ, ξ).

— Ensemble methods seem to perform less well than Logistic-Lasso in terms of SP, except for SplitLR, which outperforms Logistic-Lasso in some cases.

— Interestingly, in terms of RC, the SplitSVM methods dominate its competitors by a large margin. Thus, SplitSVM identifies better the active set S than its competitors.

— Surprisingly and in terms of CR, the SplitSVM methods dominate its competitors by a large margin. Thus, SplitSVM identifies better the active set S than its competitors.

— In terms of PR, the SplitSVM methods, followed by SplitLR, provide the best results in all parameter-combinations of (ρ, ξ).

Overall, our ensemble SplitSVM method performs well for all performance measures except for the SP measure in Scenario 1.

TABLE 4.1 Average of the performance measures of SplitSVM and its competitors, when data are generated under Scenario 1, with $\xi \in \{0.05, 0.3\}$ and $\rho \in \{0.2, 0.5, 0.8\}$.

		$\xi = 0.05$					$\xi = 0.3$				
ρ	Method	MR	SE	SP	RC	PR	MR	SE	SP	RC	PR
0.2	SplitSVM-Lasso	0.311	0.368	0.865	0.551	0.118	0.163	0.788	0.858	0.693	0.487
	SplitSVM-EN	0.303	0.372	0.872	0.802	0.073	0.157	0.728	0.891	0.918	0.378
	Logistic-Lasso	0.325	0.219	0.921	0.04	0.091	0.187	0.562	0.921	0.045	0.364
	Logistic-EN	0.32	0.327	0.872	0.058	0.086	0.178	0.578	0.926	0.079	0.372
	SplitLR-Lasso	0.313	0.286	0.904	0.127	0.097	0.159	0.625	0.932	0.228	0.38
	SplitLR-EN	0.302	0.349	0.886	0.213	0.091	0.154	0.639	0.934	0.244	0.366
0.5	SplitSVM-Lasso	0.231	0.579	0.869	0.896	0.215	0.094	0.913	0.902	0.449	0.602
	SplitSVM-EN	0.225	0.567	0.885	0.967	0.107	0.086	0.903	0.918	0.797	0.501
	Logistic-Lasso	0.251	0.46	0.9	0.027	0.093	0.111	0.751	0.943	0.032	0.328
	Logistic-EN	0.248	0.485	0.891	0.056	0.091	0.097	0.788	0.948	0.061	0.314
	SplitLR-Lasso	0.24	0.509	0.891	0.12	0.09	0.083	0.828	0.951	0.18	0.377
	SplitLR-EN	0.235	0.518	0.896	0.144	0.086	0.077	0.843	0.954	0.244	0.365
0.8	SplitSVM-Lasso	0.188	0.737	0.851	0.884	0.195	0.081	0.977	0.896	0.509	0.609
	SplitSVM-EN	0.183	0.729	0.862	0.998	0.128	0.075	0.969	0.906	0.825	0.494
	Logistic-Lasso	0.199	0.652	0.879	0.018	0.06	0.073	0.862	0.956	0.025	0.304
	Logistic-EN	0.192	0.652	0.889	0.036	0.059	0.056	0.9	0.964	0.053	0.282
	SplitLR-Lasso	0.193	0.66	0.884	0.104	0.092	0.055	0.913	0.961	0.157	0.416
	SplitLR-EN	0.187	0.672	0.888	0.138	0.086	0.05	0.922	0.963	0.248	0.385

From Table 4.2, which summarizes the results of Scenario 2, we can note that SplitSVM is very efficient in terms of three measures of performance MR, SE and RC, and its performance is consistent

through all the parameter-combinations of (ρ, ξ). In terms of PR measure, the ensemble methods dominate the traditional regularized methods, as in Scenario 1, for all configurations of (ρ, ξ). This confirms the good behavior of the SplitSVM methods obtained in Scenario 1.

TABLE 4.2 Average of the performance measures of SplitSVM and its competitors, when data are generated under Scenario 2, with $\xi \in \{0.05, 0.3\}$ and $\rho \in \{0.2, 0.5, 0.8\}$.

		$\xi = 0.05$					$\xi = 0.3$				
ρ	Method	MR	SE	SP	RC	PR	MR	SE	SP	RC	PR
0.2	SplitSVM-Lasso	0.328	0.242	0.902	0.607	0.102	0.178	0.7	0.875	0.677	0.503
	SplitSVM-EN	0.326	0.146	0.958	0.787	0.062	0.168	0.637	0.913	0.915	0.382
	Logistic-Lasso	0.342	0.133	0.938	0.071	0.278	0.216	0.472	0.918	0.092	0.781
	Logistic-EN	0.343	0.151	0.929	0.104	0.251	0.197	0.499	0.932	0.137	0.757
	SplitLR-Lasso	0.338	0.113	0.956	0.144	0.36	0.177	0.554	0.939	0.316	0.58
	SplitLR-EN	0.334	0.178	0.926	0.316	0.286	0.177	0.548	0.941	0.395	0.549
0.5	SplitSVM-Lasso	0.268	0.39	0.912	0.851	0.29	0.092	0.895	0.911	0.457	0.493
	SplitSVM-EN	0.282	0.267	0.958	0.942	0.161	0.08	0.859	0.945	0.791	0.468
	Logistic-Lasso	0.297	0.26	0.933	0.131	0.506	0.119	0.711	0.947	0.069	0.716
	Logistic-EN	0.292	0.263	0.94	0.204	0.452	0.114	0.707	0.955	0.124	0.685
	SplitLR-Lasso	0.284	0.311	0.927	0.429	0.311	0.097	0.764	0.957	0.292	0.562
	SplitLR-EN	0.285	0.276	0.945	0.391	0.304	0.099	0.747	0.96	0.337	0.562
0.8	SplitSVM-Lasso	0.198	0.664	0.873	0.838	0.146	0.078	0.931	0.921	0.4	0.496
	SplitSVM-EN	0.218	0.478	0.938	0.993	0.16	0.079	0.93	0.92	0.816	0.454
	Logistic-Lasso	0.239	0.477	0.908	0.127	0.517	0.094	0.793	0.956	0.051	0.611
	Logistic-EN	0.238	0.463	0.918	0.267	0.456	0.082	0.824	0.961	0.136	0.65
	SplitLR-Lasso	0.23	0.492	0.914	0.431	0.289	0.07	0.86	0.963	0.248	0.487
	SplitLR-EN	0.228	0.484	0.924	0.56	0.331	0.067	0.856	0.969	0.431	0.523

The results of Scenario 3 are reported in Table 4.3. We note from this table that again, in general, the ensemble methods are relatively better than the standard regularized methods in terms of MR and SE measures, except in one configuration, when $\rho = 0.2$ and $\xi = 0.05$). The results, in terms

of SP and RC, are similar to those reported from Scenarios 1 and 2. SplitLR is better than its competitors in terms of PR for small value of $\rho = 0.2$, but standard regularized approaches perform better than the ensemble methods for moderate and large values of ρ.

TABLE 4.3 Average of the performance measures of SplitSVM and its competitors, when data are generated under Scenario 3, with $\xi \in \{0.05, 0.3\}$ and $\rho \in \{0.2, 0.5, 0.8\}$.

		$\xi = 0.05$					$\xi = 0.3$				
ρ	Method	MR	SE	SP	RC	PR	MR	SE	SP	RC	PR
0.2	SplitSVM-Lasso	0.328	0.242	0.902	0.607	0.102	0.178	0.7	0.875	0.677	0.503
	SplitSVM-EN	0.326	0.146	0.958	0.787	0.062	0.168	0.637	0.913	0.915	0.382
	Logistic-Lasso	0.325	0.219	0.921	0.04	0.091	0.187	0.562	0.921	0.045	0.364
	Logistic-EN	0.32	0.327	0.872	0.058	0.086	0.178	0.578	0.926	0.079	0.372
	SplitLR-Lasso	0.338	0.113	0.956	0.144	0.36	0.177	0.554	0.939	0.316	0.58
	SplitLR-EN	0.334	0.178	0.926	0.316	0.286	0.177	0.548	0.941	0.395	0.549
0.5	SplitSVM-Lasso	0.268	0.39	0.912	0.851	0.29	0.092	0.895	0.911	0.457	0.493
	SplitSVM-EN	0.282	0.267	0.958	0.942	0.161	0.08	0.859	0.945	0.791	0.468
	Logistic-Lasso	0.284	0.315	0.924	0.127	0.341	0.115	0.741	0.941	0.072	0.7
	Logistic-EN	0.282	0.334	0.919	0.18	0.283	0.11	0.735	0.951	0.12	0.688
	SplitLR-Lasso	0.284	0.311	0.927	0.429	0.311	0.097	0.764	0.957	0.292	0.562
	SplitLR-EN	0.285	0.276	0.945	0.391	0.304	0.099	0.747	0.96	0.337	0.562
0.8	SplitSVM-Lasso	0.198	0.664	0.873	0.838	0.146	0.078	0.931	0.921	0.4	0.496
	SplitSVM-EN	0.218	0.478	0.938	0.993	0.16	0.079	0.93	0.92	0.816	0.454
	Logistic-Lasso	0.24	0.48	0.904	0.129	0.384	0.094	0.813	0.948	0.051	0.573
	Logistic-EN	0.238	0.474	0.911	0.231	0.39	0.082	0.828	0.958	0.139	0.649
	SplitLR-Lasso	0.23	0.492	0.914	0.431	0.289	0.07	0.86	0.963	0.248	0.487
	SplitLR-EN	0.228	0.484	0.924	0.56	0.331	0.067	0.856	0.969	0.431	0.523

4.4.5 The choice of the number of models

As the number of models, G, to be considered in SplitSVM is unknown, in this section, we conduct a simulation scenario to illustrate the procedure we used to determine the optimal value of G,

to integrate in the SplitSVM to report the results for Scenarios 1,2 and 3. For this illustrative simulation example, we assume that data are generated under Scenario 3, with $\rho = 0.5$, $\xi = 0.4$, and the marginal probability of the response variable $P(y_i = 1) = 0.4$. For each replication, we generated two data-sets : a training data of $n = 50$ observations and a test data with size $n = 200$. Then, we consider a grid of four values of $G = \{2, 5, 7, 10\}$, and for each value of G, we fit SplitSVM using the training data to choose the optimal values of λ_1 and λ_d, and use the test data to compute the performance measures described in Section 4.4.3. For more guidance on how to chose G, we also calculate the overlap measure OV defined as follows

$$OV = \frac{\sum_{j=1}^{p} o_j \boldsymbol{I}\{o_j \neq 0)\}}{\sum_{j=1}^{p} \boldsymbol{I}\{o_j \neq 0\}},$$

where $o_j = \frac{1}{G}\sum_{g=1}^{G} \boldsymbol{I}\{\hat{\beta}_j^g \neq 0)\}$, for each $1 \leq j \leq p$.
Note that $0 \leq OV \leq 1$, where $OV = 0$ means that all the models are null, and $OV = 1$ means that the active variables are active in all the G models. Notice also that if $o_j = 1/G$ for all $j = 1, \ldots, p$, then $OV = 1/G$, and hence each variable j is active in one model.
All the performance measures are calculated on the test data, for each fixed value of G, and for each replication data. Table 4.4 reports the average, over the $N = 30$ replications, of the six performance statistics.

From Table 4.4, one can observe that the misclassification rate MR and the overlap OV measures decrease when the number of models G increases. On the other hand, one can also notice that there is a small change on the SE and SP criteria for the different values of G. We can observe also that RC increases when G increases. Finally, when ξ increases, the latter criterion increase. Hence, taking a value of $G \geq 5$ leads to the best accuracy prediction.

TABLE 4.4 Average of the performance measures of SplitSVM at different values of the number of diversity groups, G, over 30 test sets, for two values of the sparsity parameter $\xi = 0.1$ and $\xi = 0.3$.

	$\xi = 0.05$						$\xi = 0.3$					
G	MR	SE	SP	RC	PR	OV	MR	SE	SP	RC	PR	OV
2	0.168	0.821	0.839	0.442	0.447	0.766	0.11	0.953	0.846	0.261	0.71	0.557
5	0.161	0.824	0.85	0.688	0.34	0.401	0.102	0.949	0.863	0.417	0.677	0.283
7	0.16	0.824	0.851	0.714	0.324	0.485	0.096	0.957	0.865	0.521	0.633	0.262
10	0.162	0.828	0.846	0.784	0.241	0.516	0.094	0.955	0.871	0.644	0.61	0.229

4.4.6 Real data-sets study

In this section, we evaluate the performance of our method comparing to its competitors based on analysis of real data. Therefore, to illustrate the importance of SplitSVM in terms of classification, we consider three high dimensional gene expression data sets, namely, the bisulfate DNA methylation sequencing data Kharoubi *et al.* (2019), Prostate data Singh *et al.* (2002) and Leukemia data Golub *et al.* (1999). The three data sets can be used in the context of binary classification. These data sets are described next.

The bisulfate DNA methylation data set is collected around the *BLK* gene located in chromosome 8. This data set contains measurements of DNA methylation levels of 40 different samples of cell types : B cells (8 samples), T cells (19 samples), and monocytes (13 samples). Methylation levels are measured in $p = 5896$ CpG sites (predictors).

The Prostate data, available in the R package, `spls`, consists of $n = 102$ samples ; 52 subjects with disease and 50 subjects are not. This data set contains expression of $p = 6033$ genes (predictors) across the samples.

Leukemia data set contains $n = 72$ observations and $p = 6817$ predictors. We have two types of acute leukemias : acute lymphoblastic leukemia (ALL) with 47 observations and acute myeloid leukemia (AML) with 25 observations.
A pre-screening Tibshirani *et al.* (2003) is used to choose the most (marginally) important predictors

in all data sets, using a cut-off of $p = 1000$ predictors that are retained for further analysis. Then, we split the data to training/test subsets, with 40% for training and 60% to compute the performance measures. The number of models used in the ensemble methods SplitSVM and SplitLR is choosen in the grid $G = \{2, 5, 10\}$, using cross-validation. For the methods SplitLR and Logistic, the parameter of EN is fixed to $\alpha = 0.75$. This is also the case for SplitSVM as its parameter of EN penalty, λ_2, is fixed to $\lambda_2 = 0.75$.

The whole procedure is repeated 100 times, and the average of the MR, SE and SP statistics is reported in Table 5.

TABLE 4.5 Average, over 100 runs, of the performance statistics, MR, SE and SP. Results reported for six classification methods, including SplitSVM.

	Methylation			Prostate			Leukemia		
Method	MR	SE	SP	MR	SE	SP	MR	SE	SP
SplitSVM-Lasso	0.032	0.975	0.967	0.052	0.947	0.95	0.021	0.96	0.984
SplitSVM-EN	0.056	0.825	0.967	0.055	0.938	0.953	0.024	0.96	0.981
Logistic-Lasso	0.089	0.7	0.962	0.08	0.928	0.893	0.057	0.94	0.944
Logistic-EN	0.087	0.75	0.962	0.072	0.934	0.89	0.04	0.94	0.966
SplitLR-Lasso	0.059	0.857	0.957	0.069	0.934	0.927	0.033	0.96	0.969
SplitLR-EN	0.053	0.893	0.957	0.069	0.925	0.937	0.033	0.96	0.969

From Table 4.5, one can clearly see that SplitSVM performs better than its competitors according to all the three measures of performance MR, SE and SP.

4.5 Discussion

We have proposed SplitSVM, a new ensemble approach in binary classification based on Support Vector Machine classifier. SplitSVM is based on optimizing an objective function that smoothly approximates SVM hinge loss function, and it is regularized using two penalties. The Lasso penalty is used to obtain sparse solutions in each model of the ensemble, and the diversity penalty is exploited in order to build divergent models based on estimation of different active groups/models of predictors.

We derive an upper bound of the error of SplitSVM, and we show that this error achieves the $\boldsymbol{O}(\sqrt{\frac{s\log(p)}{n}})$ rate.

The simulation study and real data analysis show that our ensemble method, SplitSVM, does better than the regularized SVM and existent methods, such as SplitLR and Logistic regression.

In this work, we have focused on Lasso penalty to reach sparsity, however, one can replace Lasso with a non-convex penalty, such as SCAD Fan et Li (2001) or MCP Zhang (2010). It is known in the literature that such penalties reduce the bias of the model parameter estimators. This avenue can be investigated in future work.

4.6 Appendix

4.6.1 Appendix **A**

*Proof. (***Proof of Theorem 4.3***)*

For the proof of Theorem 4.3, we first we prove that the error $\boldsymbol{h} = \hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star$, belongs to the set $C(S)$ and second, we derive an upper bound of the difference $\boldsymbol{h}$, for each $g = 1, \ldots, G$.

We have $L(\hat{\boldsymbol{\beta}}) \leq L(\boldsymbol{\beta}^\star)$, which implies that $L(\boldsymbol{h} + \boldsymbol{\beta}^\star) \leq L(\boldsymbol{\beta}^\star)$. Then

$$\frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i \boldsymbol{x}_i^\top(\boldsymbol{h}+\boldsymbol{\beta}^\star)) + \lambda_1||\boldsymbol{h}+\boldsymbol{\beta}^\star||_1 + \frac{\lambda_d}{2}||\hat{\boldsymbol{W}}(\boldsymbol{h}+\boldsymbol{\beta}^\star)||_1 \leq \frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star)) + \lambda_1||\boldsymbol{\beta}^\star||_1 + \frac{\lambda_d}{2}||\hat{\boldsymbol{W}}\boldsymbol{\beta}^\star||_1,$$

it follows that

$$\frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i \boldsymbol{x}_i^\top(\boldsymbol{h}+\boldsymbol{\beta}^\star)) - \frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star)) \leq \lambda_1(||\boldsymbol{\beta}^\star||_1 - ||\boldsymbol{h}+\boldsymbol{\beta}^\star||_1) + \frac{\lambda_d}{2}(||\hat{\boldsymbol{W}}\boldsymbol{\beta}^\star||_1 - ||\hat{\boldsymbol{W}}(\boldsymbol{h}+\boldsymbol{\beta}^\star)||_1). \tag{4.10}$$

Notice that the BernSVM loss function is twice differentiable. Thus, we apply a second order Taylor expansion to $B_\delta(t)$, which gives

$$\frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i \boldsymbol{x}_i^\top(\boldsymbol{h}+\boldsymbol{\beta}^\star)) - \frac{1}{n}\sum_{i=1}^{n} B_\delta(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star)) = \frac{1}{n}\sum_{i=1}^{n} B'_\delta(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star) y_i \boldsymbol{x}_i^\top \boldsymbol{h} + \frac{\boldsymbol{h}^\top \boldsymbol{H}\boldsymbol{h}}{2}, \tag{4.11}$$

where $\boldsymbol{H} = \boldsymbol{X}^\top \boldsymbol{D}\boldsymbol{X}/n$, with $\boldsymbol{D} = diag(B'_\delta(y_i \boldsymbol{x}_i^\top \boldsymbol{\beta}^\star))$, for $i = 1, \ldots, n$.

We can see also that

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} -B_{\delta}^{'}(y_i \boldsymbol{x}_i^{\top}\boldsymbol{\beta}^{\star}) y_i \boldsymbol{x}_i^{\top}\boldsymbol{h} &\leq \frac{1}{n}\sum_{i=1}^{n} |B_{\delta}^{'}(y_i \boldsymbol{x}_i^{\top}\boldsymbol{\beta}^{\star}) y_i \boldsymbol{x}_i^{\top}\boldsymbol{h}| \\
&\leq \frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{x}_i^{\top}\boldsymbol{h}| \quad \textit{because} \quad |B_{\delta}^{'}(y_i \boldsymbol{x}_i^{\top}\boldsymbol{\beta}^{\star}) y_i| = |B_{\delta}^{'}(y_i \boldsymbol{x}_i^{\top}\boldsymbol{\beta}^{\star})| \leq 1 \\
&= \frac{1}{n}\sum_{i=1}^{n} |\sum_{j=1}^{p} x_{ij} h_j| \\
&\leq \sum_{j=1}^{p} (\frac{1}{n}\sum_{i=1}^{n} |x_{ij}|)|h_j| \\
&\leq \sum_{j=1}^{p} (\frac{1}{\sqrt{n}} \sqrt{\sum_{i=1}^{n} |x_{ij}|^2})|h_j| \\
&\leq \frac{M}{\sqrt{n}} ||\boldsymbol{h}||_1 \quad \textit{because} \quad ||\boldsymbol{x}_j||_2 \leq M \quad \forall j.
\end{aligned}$$

Then, using Equations (4.10) and (4.11), we obtain

$$\begin{aligned}
\frac{\boldsymbol{h}^{\top}\boldsymbol{H}\boldsymbol{h}}{2} &\leq \frac{1}{n}\sum_{i=1}^{n} -B_{\delta}^{'}(y_i \boldsymbol{x}_i^{\top}\boldsymbol{\beta}^{\star}) y_i \boldsymbol{x}_i^{\top}\boldsymbol{h} + \lambda_1(||\boldsymbol{\beta}^{\star}||_1 - ||\boldsymbol{h}+\boldsymbol{\beta}^{\star}||_1) + \frac{\lambda_d}{2}(||\hat{\boldsymbol{W}}\boldsymbol{\beta}^{\star}||_1 - ||\hat{\boldsymbol{W}}(\boldsymbol{h}+\boldsymbol{\beta}^{\star})||_1) \\
&\leq \frac{M}{\sqrt{n}}||\boldsymbol{h}||_1 + \lambda_1(||\boldsymbol{\beta}^{\star}||_1 - ||\boldsymbol{h}+\boldsymbol{\beta}^{\star}||_1) + \frac{\lambda_d}{2}(||\hat{\boldsymbol{W}}\boldsymbol{\beta}^{\star}||_1 - ||\hat{\boldsymbol{W}}(\boldsymbol{h}+\boldsymbol{\beta}^{\star})||_1).
\end{aligned}$$

Moreover, we have

$$\begin{aligned}
||\hat{\boldsymbol{W}}\boldsymbol{\beta}^{\star}||_1 - ||\hat{\boldsymbol{W}}(\boldsymbol{h}+\boldsymbol{\beta}^{\star})||_1 &= ||\hat{\boldsymbol{W}}_S\boldsymbol{\beta}_S^{\star}||_1 - ||\hat{\boldsymbol{W}}_S(\boldsymbol{h}_S+\boldsymbol{\beta}_S^{\star})||_1 - ||\hat{\boldsymbol{W}}_{S^c}\boldsymbol{h}_{S^c}||_1 \\
&\leq ||\hat{\boldsymbol{W}}_S\boldsymbol{h}_S||_1 - ||\hat{\boldsymbol{W}}_{S^c}\boldsymbol{h}_{S^c}||_1 \\
&\leq ||\boldsymbol{W}_S\boldsymbol{h}_S||_1 - ||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1,
\end{aligned}$$

where we have used in the first equality $\boldsymbol{\beta}_{S^c}^{\star} = \mathbf{0}$ *and the first inequality is based on the fact that* $|a|-|b| \leq |a-b|$*, for all* $a, b \in \mathbb{R}$*. The second inequality is due to the fact that the event* $\Omega_0 = \{\hat{w}_j \leq$

$w_j, \forall j \in S\} \cup \{w_j \leq \hat{w}_j, \forall j \in S^c\}$ *is realized. Then we obtain*

$$\begin{aligned} 0 &< \frac{\boldsymbol{h}^\top \boldsymbol{H}\boldsymbol{h}}{2} \\ &\leq \frac{M}{\sqrt{n}}||\boldsymbol{h}||_1 + \lambda_1||\boldsymbol{h}||_1 + \frac{\lambda_d}{2}(||\boldsymbol{W}_S\boldsymbol{h}_S||_1 - ||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1) \\ &= (\frac{M}{\sqrt{n}} + \lambda_1)||\boldsymbol{h}||_1 + \frac{\lambda_d}{2}(||\boldsymbol{W}_S\boldsymbol{h}_S||_1 - ||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1). \end{aligned}$$

Since the right hand of the last inequality is positive, we obtain with probability $1 - \exp(-c_0 n)$

$$\begin{aligned} 0 &\leq (\frac{M}{\sqrt{n}} + \lambda_1)||\boldsymbol{h}||_1 + \frac{\lambda_d}{2}(||\boldsymbol{W}_S\boldsymbol{h}_S||_1 - ||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1) \\ &= (\frac{M}{\sqrt{n}} + \lambda_1)||\boldsymbol{h}_S||_1 + (\frac{M}{\sqrt{n}} + \lambda_1)||\boldsymbol{h}_{S^c}||_1 + \frac{\lambda_d}{2}(||\boldsymbol{W}_S\boldsymbol{h}_S||_1 - ||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1) \\ &= (\frac{M}{\sqrt{n}} + \lambda_1)||\boldsymbol{h}_S||_1 + (\frac{M}{\sqrt{n}} + \lambda_1)||\boldsymbol{W}_{S^c}^{-1}\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1 + \frac{\lambda_d}{2}(||\boldsymbol{W}_S\boldsymbol{h}_S||_1 - ||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1) \\ &\leq (\frac{M}{\sqrt{n}} + \lambda_1)||\boldsymbol{h}_S||_1 + (\frac{M}{\sqrt{n}} + \lambda_1)||\boldsymbol{W}_{S^c}^{-1}||_\infty||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1 + \frac{\lambda_d}{2}||\boldsymbol{W}_S||_\infty||\boldsymbol{h}_S||_1 - \frac{\lambda_d}{2}||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1) \\ &= (\frac{M}{\sqrt{n}} + \lambda_1 + \frac{\lambda_d}{2}||\boldsymbol{W}_S||_\infty)||\boldsymbol{h}_S||_1 - \left[\frac{\lambda_d}{2} - (\frac{M}{\sqrt{n}} + \lambda_1)||\boldsymbol{W}_{S^c}^{-1}||_\infty\right]||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1. \end{aligned}$$

It follows that the Lasso BernSVM error satisfies the cone constraint given by

$$\mathcal{C}(S) = \{\boldsymbol{h} \in \mathbb{R}^p : ||\boldsymbol{W}_{S^c}\boldsymbol{h}_{S^c}||_1 \leq \gamma||\boldsymbol{h}_S||_1\},$$

where

$$\gamma < \frac{\frac{M}{\sqrt{n}} + \lambda_1 + \frac{\lambda_d}{2}||\boldsymbol{W}_S||_\infty)}{\frac{\lambda_d}{2} - (\frac{M}{\sqrt{n}} + \lambda_1)||\boldsymbol{W}_{S^c}^{-1}||_\infty}.$$

Thus, we prove that the error $\boldsymbol{h}$ *belongs to the set* $\mathcal{C}(S)$.

We state here the definition of the Restricted Strong Convexity (RSC) condition.

Definition 4.5 *A design matrix* $\boldsymbol{X}$ *satisfies the RSC condition on* $\mathcal{C}$ *with* $\kappa > 0$ *if*

$$\frac{||\boldsymbol{X}\boldsymbol{h}||_2^2}{n} \geq \kappa||\boldsymbol{h}||_2^2, \quad \forall \boldsymbol{h} \in \mathcal{C}.$$

The next lemma Kharoubi et al. *(2023) shows that the Hessian matrix satisfies the RSC condition, for a specific n, the number of observations in the design matrix $\boldsymbol{X}$.*

Lemma 4.6 *If the assumptions A1 and A2 are satisfied and $n = \Omega(s\log(p))$, then $\boldsymbol{H}$ satisfies the RSC with $\kappa = \kappa_1\kappa_3/2$.*

Furthermore, from the inequality above and Lemma 4.6 we have

$$\begin{aligned}
\kappa||\boldsymbol{h}||_2^2 &\leq \left(\frac{M}{\sqrt{n}} + \lambda_1 + \frac{\lambda_d}{2}||\boldsymbol{W}_S||_\infty\right)||\boldsymbol{h}_S||_1 - \left(\frac{\lambda_d}{2} - (\frac{M}{\sqrt{n}} + \lambda_1)||\boldsymbol{W}_{S^c}^{-1}||_\infty\right)||\boldsymbol{h}_{S^c}||_1 \\
&\leq \left(\frac{M}{\sqrt{n}} + \lambda_1 + \frac{\lambda_d}{2}||\boldsymbol{W}_S||_\infty\right)||\boldsymbol{h}_S||_1 \\
&\leq \left(\frac{M}{\sqrt{n}} + \lambda_1 + \frac{\lambda_d}{2}||\boldsymbol{W}_S||_\infty\right)\sqrt{s}||\boldsymbol{h}||_2.
\end{aligned}$$

Thus, we obtain an upper bound of the error $\boldsymbol{h} = \hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star$ given by

$$||\boldsymbol{h}||_2 \leq \frac{(\frac{M}{\sqrt{n}} + \lambda_1 + \frac{\lambda_d}{2}||\boldsymbol{W}_S||_\infty)\sqrt{s}}{\kappa}.$$

Finally, we deduce an upper bound of the L_2 error $\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star$.

In fact, we have

$$\begin{aligned}
||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2^2 &= ||\frac{1}{G}\sum_{g=1}^{G}(\hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star)||_2^2 \\
&\leq \frac{1}{G^2}||\sum_{g=1}^{G}(\hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star)||_2^2 \\
&\leq \frac{1}{G^2}(\sum_{g=1}^{G}||\hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star||_2)^2 \\
&\leq \frac{1}{G}\sum_{g=1}^{G}||\hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star||_2^2 \\
&\leq \frac{1}{G}\sum_{g=1}^{G}||\hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star||_2^2 \\
&\leq (\frac{M}{\sqrt{n}} + \lambda_1 + \frac{\lambda_d}{2}||\boldsymbol{W}_S||_\infty)\sqrt{s}||\boldsymbol{h}||_2.
\end{aligned}$$

Thus, we conclude that

$$||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^\star||_2 \leq (\frac{M}{\sqrt{n}} + \lambda_1 + \frac{\lambda_d}{2}||\boldsymbol{W}_S||_\infty)\sqrt{s}.$$

If we take λ_1 and λ_d to be of order $\sqrt{\log(p/n)}$, we obtain an upper bound of order $\sqrt{s\log(p/n)}$. □

4.6.2 Appendix **B**

*Proof. (***Proof of Theorem 4.4***)*

We have

$$\begin{aligned}
\mathcal{E}(f_{\hat{\boldsymbol{\beta}}^g}) &= R(f_{\hat{\boldsymbol{\beta}}^g}) - R(f^\star) \\
&= R(f_{\hat{\boldsymbol{\beta}}^g}) - R(f_{\boldsymbol{\beta}^\star}) + R(f_{\boldsymbol{\beta}^\star}) - R(f^\star) \\
&= \mathcal{E}(f_{\boldsymbol{\beta}^\star}) + R(f_{\hat{\boldsymbol{\beta}}^g}) - R(f_{\boldsymbol{\beta}^\star}).
\end{aligned}$$

The BernSVM loss function has a second derivative continuous and bounded, then using a second Taylor expansion, we obtain

$$R(f_{\hat{\boldsymbol{\beta}}^g}) - R(f_{\boldsymbol{\beta}^\star}) \leq \frac{L}{2}||f_{\hat{\boldsymbol{\beta}}^g} - f_{\boldsymbol{\beta}^\star}||_2^2.$$

Hence, one has

$$\mathcal{E}(f_{\hat{\boldsymbol{\beta}}^g}) \leq \mathcal{E}(f_{\boldsymbol{\beta}^\star}) + \frac{L}{2}||f_{\hat{\boldsymbol{\beta}}^g} - f_{\boldsymbol{\beta}^\star}||_2^2.$$

We have $f_{\hat{\boldsymbol{\beta}}^g} = \boldsymbol{X}\hat{\boldsymbol{\beta}}^g$ and $f_{\boldsymbol{\beta}^\star} = \boldsymbol{X}\boldsymbol{\beta}^\star$, then we obtain

$$\begin{aligned}
||f_{\hat{\boldsymbol{\beta}}^g} - f_{\boldsymbol{\beta}^\star}||_2^2 &= ||\boldsymbol{X}\hat{\boldsymbol{\beta}}^g - \boldsymbol{X}\boldsymbol{\beta}^\star||_2^2 \\
&= ||\boldsymbol{X}(\hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star)||_2^2 \\
&\leq ||\boldsymbol{X}||_2^2||\hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star||_2^2 \\
&\leq \Lambda_{\max} n||\hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star||_2^2.
\end{aligned}$$

Thus, it follows that

$$\frac{1}{G}\sum_{g=1}^{G}\mathcal{E}(f_{\hat{\boldsymbol{\beta}}^g}) \leq \mathcal{E}(f_{\boldsymbol{\beta}^\star}) + \frac{L}{2G}\Lambda_{\max} n\sum_{g=1}^{G}||\hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star||_2^2.$$

Using the convexity of the BernSVM loss function we obtain

$$\frac{1}{G}\sum_{g=1}^{G}\mathcal{E}(f_{\hat{\boldsymbol{\beta}}^g}) = \frac{1}{G}\sum_{g=1}^{G}R(f_{\hat{\boldsymbol{\beta}}^g}) - R(f_{\boldsymbol{\beta}^\star}) \geq R(\frac{1}{G}\sum_{g=1}^{G}f_{\hat{\boldsymbol{\beta}}^g}) - R(f_{\boldsymbol{\beta}^\star}).$$

Thus, we get

$$R(\frac{1}{G}\sum_{g=1}^{G}f_{\hat{\boldsymbol{\beta}}^g}) - R(f_{\boldsymbol{\beta}^\star}) \leq \mathcal{E}(f_{\boldsymbol{\beta}^\star}) + \frac{L}{2G}\Lambda_{\max}n\sum_{g=1}^{G}||\hat{\boldsymbol{\beta}}^g - \boldsymbol{\beta}^\star||_2^2.$$

If we have $f_{\boldsymbol{\beta}^\star} = f^\star$ *and* $\Lambda_{\max} = \boldsymbol{o}(\frac{1}{s\log p})$*, then* $\frac{1}{G}\sum_{g=1}^{G}f_{\hat{\boldsymbol{\beta}}^g}$ *is consistent.* □

CONCLUSION

Cette thèse s'inscrit dans le cadre de l'apprentissage statistique supervisé. En effet, lorsque nous avons un ensemble d'observations avec une variable réponse binaire y et une matrice de prédicteurs $\boldsymbol{x}$, l'objectif généralement est de modéliser la relation entre y et $\boldsymbol{x}$ et généraliser cette relation à l'échelle de la population pour faire de la prévision et le classement binaire. Cependant, avec l'avancée de la science, les données de grandes dimensions où le nombre de prédicteurs est beaucoup plus grand que le nombre d'observations, sont de plus en plus collectées. Pour analyser ce type de données complexes, nous cherchons toujours des modèles parcimonieux et faciles à intérpréter, avec une bonne capacité prédictive. Bien que la méthode SVM standard a un pouvoir prédictif important, elle n'est pas bien adaptée pour analyser de telles données. D'où la nécessité des méthodes de régularisations, comme Lasso, Elastic-Net et autres, afin d'améliorer la performance de la méthode SVM pour analyser les données de grandes dimensions.

L'objectif de ces travaux de recherche est d'adapter le classificateur SVM dans ce cadre de données. En effet, la thèse propose trois extensions de la méthode SVM capables d'analyser les données de grandes dimensions de façon appropriée, en préservant toujours le principe de parcimonie des modèles sous-jaçents et en assurant une bonne performance au niveau de la prédiction/classement binaire.

Dans le chapitre 2, nous avons proposé l'approche CCNSVM. C'est une méthode SVM basée sur une nouvelle pénalité appellée CCN. Cette dérnière est capable de capturer le regroupement des prédicteurs qui sont importantes pour la variable réponse. En effet, nous avons exploité la structure de corrélation entre les prédicteurs pour calculer une matrice de simularité TOM. Ensuite, nous avons exploité les colones de cette dérnière afin de construire la pénalité CCN. Nous avons proposé un algorithme de type minimisation par alternance dans lequel, nous avons estimé d'abord la structure de groupes des prédicteurs et ensuite nous avons estimé les coefficients des paramètres du modèle SVM pénalisé. La méthode CCNSVM favorise le regroupement des variables importantes pour la variable réponse et améliore la prédiction/classement. L'analyse des données simulées et des données réelles a démontré l'utilité de cette approche, surtout pour le traitement des données pour lequelles il y avait une structure de groupe dans les prédicteurs.

Dans le chapitre 3, notre objectif a été d'élaborer des propriétés théoriques, non asymptotiques de la méthode SVM régularisée. Cependant, la fonction de perte SVM ne possède pas de dérivée seconde. En plus, son approximation, déjà existante dans la littérature, la fonction de perte Huber, ne l'est pas également. Par conséquent, nous nous sommes intéressés à l'approximation de la fonction de perte SVM par une fonction de perte lisse et suffisamment dérivable pour faciliter l'atteinte des propriétés théoriques du classificateur SVM. En effet, nous avons utilisé les polynômes de Bernstein pour approximer la fonction de perte SVM. La fonction de perte obtenue, nommée BernSVM, est deux fois dérivable et sa dérivée seconde est continue. Pour illustrer l'importance de la fonction de perte BernSVM, nous avons proposé deux algorithmes pour résoudre le problème d'optimisation de la méthode BernSVM pénalisée. Le premier algorithme, appelé BSVM-GCD, combine la descente par coordonnée et la technique de maximisation-minimisation. En effet, comme la dérivée seconde de BernSVM est bornée, ceci nous a permis de la majorer par une fonction quadratique facile à minimiser en utilisant la descente par coordonnée. Le deuxième algorithme exploite la propriété de l'existence et de la continuité de la derivée seconde de la fonction de perte BernSVM pour résoudre le problème d'optimisation sous-jaçent via une technique efficace des moindres carrés itérative. Cet algorithme est nommé BSVM-IRLS. Nous avons menés des études de simulation pour illustrer la rapidité d'estimation de BSVM-IRLS. D'un point de vue théorique, nous avons utilisé les propriétés de la fonction de perte BernSVM pour satisfaire aux conditions théoriques nécessaires pour dériver des propriétés non asymptotiques de la méthode BernSVM en grande dimension. Plus précisément, nous avons proposé une borne supérieure pour la différence entre les estimateurs des coefficients des paramètres de la méthode BernSVM pénalisée et le vrai vecteur des coefficients du modèle proposé. La borne obtenue est un $\boldsymbol{O}_P(\sqrt{s\log(p)/n})$. Les scénarios de simulations et l'analyse de données réelles ont montré l'importance et l'efficacité de la méthode BernSVM comparée à des approches compétiteurs.

Dans le chapitre 4, nous nous sommes intéressés aux méthodes ensemblistes. Ces méthodes sont définies comme un ensemble de classificateurs qui proviennent de différents modèles ou d'un même modèle et dont les décisions de prédiction sont prises par agrégation de l'information capturée dans les données par chacun de ces modèles. Ainsi, elles ont suscités beaucoup d'intérêt pour le traitement de données de grande dimension. Alors, nous avons proposé la méthode SplitSVM, une nouvelle approche de la théorie des méthodes ensemblistes basée sur l'aggrégation de plusieurs modèles de type BernSVM pénalisée. La méthode SplitSVM est proposée afin de remédier au problème

d'interprétabilité des modèles et des résultats dont souffrent les méthodes ensemblistes standards. La méthode SplitSVM est basée sur une combinaison linéaire de deux pénalités. La pénalité Lasso est utilisée pour obtenir des modèles parcimonieux et une pénalité de diversité pour obtenir des modèles différents, voire même disjoints. Nous avons élaboré aussi une propriété non asymptotique des estimateurs des coefficients des paramètres du modèle SplitSVM. En effet, nous avons obtenu une borne supérieure pour l'erreur L_2 de la différence entre les estimations des coefficients des paramètres de la méthode SplitSVM et le vecteur des coefficients du vrai modèle. Les études de simulation menées et l'analyse de données réelles ont démontré l'utilité de la méthode SplitSVM pour le classement binaire.

Les méthodes proposées dans cette thèse peuvent être étendus dans plusieurs directions. Dans un premier lieu, nous avons utilisé la pénalité Lasso comme méthode de régularisation pour obtenir des modèles parcimonieux. Cependant, nous pouvons utiliser d'autres pénalités convexes comme Elastci-Net ou des pénalités non convexes comme SCAD et MCP. Nous pouvons également adapter la méthodologie de la thèse pour les pénalités de groupes comme, entre autres, groupe Lasso ou groupe SCAD (Yang et Zou, 2015).

Une autre nouvelle direction à explorer dans mes traveaux de recherche pourrait être la selection des variables importantes lorsque ces dernières sont fortement corrélées, dans le cadre des données de grande dimension. Ainsi, je pourrais d'abord s'adresser aux problèmes de la forte corrélation en proposant une méthode pour décorréler les prédicteurs qui sont fortement corrélées (Fan *et al.*, 2013). En effet, je pourrais utiliser une analyse en composantes principales pour transformer la matrice de données $\boldsymbol{X}$ en une matrice composée par des nouveaux prédicteurs (facteurs) non corrélés. Par la suite, je pourrais appliquer la méthode BernSVM pénalisée, présentée au chapitre 3, pour estimer les coefficients des paramètres du modèle SVM. Dans le cadre de la régression linéaire sparse lorsque les données présentent une dépendence sérielle, par exemple une dépendence temporelle ou spatiale, (Fan *et al.*, 2020) affirment que cette technique remédie à l'effet des corrélations fortes entre les prédicteurs. La fonction de perte BernSVM est suffisament lisse et pourrait satisfaire ainsi aux conditions théoriques nécessaires pour prouver la consistence de cette nouvelle méthode. Par conséquent, cette méhode serait intéressante pour améliorer le classement binaire en grande dimension en présence de forte corrélation entre les prédicteurs.

BIBLIOGRAPHIE

Anzarmou, Y. et Mkhadri, A. (2022). Classification in high dimension with an alternative feature augmentation via nonparametrics and selection (afans). *Communication in Statistics-Computation and Simulation*, p. https ://doi.org/10.1080/03610918.2021.2024232.

Anzarmou, Y., Mkhadri, A. et Oualkacha, K. (2023). Sparse overlapped linear discriminant analysis. *Test, 23, 388-417.*

Bach, F. R. (2008). Bolasso : model consistent lasso estimation through the bootstrap. Dans *Proceedings of the 25th international conference on Machine learning*, 33–40.

Becker, N., Toedt, G., Lichter, P. et Benner, A. (2011). Elastic scad as a novel penalization method for svm classification tasks in high-dimensional data. *BMC bioinformatics*, *12*(1), 1–13.

Bradley, P. S. et Mangasarian, O. L. (1998). Feature selection via concave minimization and support vector machines. Dans *ICML*, volume 98, 82–90. Citeseer.

Breiman, L. (1996). *Bias, variance, and arcing classifiers.* Rapport technique, Tech. Rep. 460, Statistics Department, University of California, Berkeley ….

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Bühlmann, P. et Van De Geer, S. (2011). *Statistics for high-dimensional data : methods, theory and applications.* Springer Science & Business Media.

Cai, T. et Liu, W. (2011). A direct estimation approach to sparse linear discriminant analysis. *Journal of the American statistical association*, *106*(496), 1566–1577.

Chang, H. H. et Chen, S. W. (2008). The impact of online store environment cues on purchase intention : Trust and perceived risk as a mediator. *Online information review.*

CHARRAD, M.-GHAZZALI, B. N., NIKNAFS, V. et CHARRAD, A. (2014). M. m.(2014). package 'nbclust'. *Journal of statistical software*, (61), 1–36.

Christidis, A.-A., Lakshmanan, L., Smucler, E. et Zamar, R. (2020). Split regularized regression. *Technometrics*, *62*(3), 330–338.

Christidis, A.-A., Van Aelst, S. et Zamar, R. (2021). Data-driven diverse logistic regression ensembles. *arXiv preprint arXiv :2102.08591*.

Cortes, C. et Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Dedieu, A. (2019). Error bounds for sparse classifiers in high-dimensions. Dans *The 22nd International Conference on Artificial Intelligence and Statistics*, 48–56. PMLR.

Fan, J., Feng, Y. et Tong, X. (2012). A road to classification in high dimensional space : the regularized optimal affine discriminant. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, *74*(4), 745–771.

Fan, J., Ke, Y. et Wang, K. (2020). Factor-adjusted regularized model selection. *Journal of Econometrics*, *216*(1), 71–85.

Fan, J. et Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, *96*(456), 1348–1360.

Fan, J., Liao, Y. et Mincheva, M. (2013). Large covariance estimation by thresholding principal orthogonal complements. *Journal of the Royal Statistical Society. Series B, Statistical methodology*, *75*(4).

Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, *27*(8), 861–874.

Freund, Y., Schapire, R. E. *et al.* (1996). Experiments with a new boosting algorithm. Dans *icml*, volume 96, 148–156. Citeseer.

Friedman, J., Hastie, T. et Tibshirani, R. (2000). Additive logistic regression : a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, *28*(2), 337–407.

Friedman, J., Hastie, T. et Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, *33*(1), 1.

Friedman, J. H. (2001). Greedy function approximation : a gradient boosting machine. *Annals of statistics*, 1189–1232.

Gareth, J., Daniela, W., Trevor, H. et Robert, T. (2013). *An introduction to statistical learning : with applications in R*. Spinger.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A. *et al.* (1999). Molecular classification of cancer : class discovery and class prediction by gene expression monitoring. *science*, *286*(5439), 531–537.

Gunn, S. R. *et al.* (1998). Support vector machines for classification and regression. *ISIS technical report*, *14*(1), 5–16.

Hardt, M. (2014). Understanding alternating minimization for matrix completion. Dans *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, 651–660. IEEE.

Hartigan, J. A. et Wong, M. A. (1979). Algorithm as 136 : A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, *28*(1), 100–108.

Hastie, T., Tibshirani, R., Friedman, J. H. et Friedman, J. H. (2009). *The elements of statistical learning : data mining, inference, and prediction*, volume 2. Springer.

Hertz, G. Z. et Stormo, G. D. (1999). Identifying dna and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics (Oxford, England)*, *15*(7), 563–577.

Hosmer Jr, D. W., Lemeshow, S. et Sturdivant, R. X. (2013). *Applied logistic regression*, volume 398. John Wiley & Sons.

Huang, J. et Zhang, C.-H. (2012). Estimation and selection via absolute penalized convex minimization and its multistage adaptive applications. *The Journal of Machine Learning Research*, *13*(1), 1839–1864.

Hunter, D. R. et Lange, K. (2004). A tutorial on mm algorithms. *The American Statistician*, *58*(1), 30–37.

Kharoubi, R., Mkhadri, A. et Oualkacha, K. (2023). High-dimensional penalized bernstein support vector machines.

Kharoubi, R., Oualkacha, K. et Mkhadri, A. (2019). The cluster correlation-network support vector machine for high-dimensional binary classification. *Journal of Statistical Computation and Simulation*, *89*(6), 1020–1043.

Koo, J.-Y., Lee, Y., Kim, Y. et Park, C. (2008). A bahadur representation of the linear support vector machine. *The Journal of Machine Learning Research*, *9*, 1343–1368.

Lakhal-Chaieb, L., Greenwood, C. M., Ouhourane, M., Zhao, K., Abdous, B. et Oualkacha, K. (2017). A smoothed em-algorithm for dna methylation profiles from sequencing-based methods in cell lines or for a single cell type. *Statistical applications in genetics and molecular biology*, *16*(5-6), 313–331.

Langfelder, P. et Horvath, S. (2014). Tutorials for the wgcna package. *UCLA. Los Ageles*.

Lee, Y.-J. et Mangasarian, O. L. (2001). Rsvm : Reduced support vector machines. Dans *Proceedings of the 2001 SIAM International Conference on Data Mining*, 1–17. SIAM.

Likas, A., Vlassis, N. et Verbeek, J. J. (2003). The global k-means clustering algorithm. *Pattern recognition*, *36*(2), 451–461.

Liu, D., Qian, H., Dai, G. et Zhang, Z. (2013). An iterative svm approach to feature selection and classification in high-dimensional datasets. *Pattern Recognition*, *46*(9), 2531–2537.

Lorentz, G. G. (2013). *Bernstein polynomials*. American Mathematical Soc.

Mai, Q., Zou, H. et Yuan, M. (2012). A direct approach to sparse discriminant analysis in ultra-high dimensions. *Biometrika*, *99*(1), 29–42.

McCullagh, P. et Nelder, J. (1989). Binary data. In *Generalized linear models* 98–148. Springer.

Meinshausen, N. et Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, *72*(4), 417–473.

Mkhadri, A., Ouhourane, M. et Oualkacha, K. (2017). A coordinate descent algorithm for computing penalized smooth quantile regression. *Statistics and Computing*, *27*(4), 865–883.

Negahban, S., Yu, B., Wainwright, M. J. et Ravikumar, P. (2009). A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. *Advances in neural information processing systems*, *22*.

Nguyen, T. et Sanner, S. (2013). Algorithms for direct 0–1 loss optimization in binary classification. Dans *International Conference on Machine Learning*, 1085–1093. PMLR.

Peng, B., Wang, L. et Wu, Y. (2016). An error bound for l1-norm support vector machine coefficients in ultra-high dimension. *The Journal of Machine Learning Research*, *17*(1), 8279–8304.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, *66*(336), 846–850.

Raskutti, G., Wainwright, M. J. et Yu, B. (2010). Restricted eigenvalue properties for correlated gaussian designs. *The Journal of Machine Learning Research*, *11*, 2241–2259.

Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N. et Barabási, A.-L. (2002). Hierarchical organization of modularity in metabolic networks. *science*, *297*(5586), 1551–1555.

Rosseeuw, P. et Van Zomeren, B. (1990). Unmasking multivariate outliers and leverate points. *Journal of the American Statistical Association*, *85*, 633–639.

Rosset, S. et Zhu, J. (2007). Piecewise linear regularized solution paths. *The Annals of Statistics*, 1012–1030.

Rudelson, M. et Zhou, S. (2012). Reconstruction from anisotropic random measurements. Dans *Conference on Learning Theory*, 10–1. JMLR Workshop and Conference Proceedings.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206–215.

Shao, J., Wang, Y., Deng, X. et Wang, S. (2011). Sparse linear discriminant analysis by thresholding for high dimensional data. *The Annals of statistics*, *39*(2), 1241–1265.

Singh, D., Febbo, P. G., Ross, K., Jackson, D. G., Manola, J., Ladd, C., Tamayo, P., Renshaw, A. A., D'Amico, A. V., Richie, J. P. *et al.* (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer cell*, *1*(2), 203–209.

Storey, J. D. et Tibshirani, R. (2003). Statistical significance for genomewide studies. *Proceedings of the National Academy of Sciences*, *100*(16), 9440–9445.

Tan, M., Wang, L. et Tsang, I. W. (2010). Learning sparse svm for feature selection on very high dimensional datasets. Dans *ICML*.

Tibshirani, R., Hastie, T., Narasimhan, B. et Chu, G. (2003). Class prediction by nearest shrunken centroids, with applications to dna microarrays. *Statistical Science*, 104–117.

Tibshirani, R., Walther, G. et Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, *63*(2), 411–423.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, *109*(3), 475–494.

Turgeon, Maxime and Oualkacha, Karim and Ciampi, Antonio and Miftah, Hanane and Dehghan, Golsa and Zanke, Brent W and Benedet, Andréa L and Rosa-Neto, Pedro and Greenwood, Celia MT and Labbe, Aurélie and others (2018). Principal component of explained variance : an efficient and optimal data dimension reduction framework for association studies. *Statistical methods in medical research*, *27*(5), 1331–1350.

Vapnik, V. (1999). *The nature of statistical learning theory*. Springer science & business media.

Wang, L., Zhu, J. et Zou, H. (2006). The doubly regularized support vector machine. *Statistica Sinica*, 589–615.

Wang, S., Song, P. X. et Zhu, J. (2010). Doubly regularized reml for estimation and selection of fixed and random effects in linear mixed-effects models.

Wendelberger, L. J., Reich, B. J. et Wilson, A. G. (2020). Multi-model penalized regression. *arXiv preprint arXiv :2006.09157*.

Witten, D. M., Shojaie, A. et Zhang, F. (2014). The cluster elastic net for high-dimensional regression with unknown variable grouping. *Technometrics*, *56*(1), 112–122.

Yang, Y. et Zou, H. (2013). An efficient algorithm for computing the hhsvm and its generalizations. *Journal of Computational and Graphical Statistics*, *22*(2), 396–415.

Yang, Y. et Zou, H. (2015). A fast unified algorithm for solving group-lasso penalize learning problems. *Statistics and Computing*, *25*(6), 1129–1141.

Ye, Y. et Godzik, A. (2004). Fatcat : a web server for flexible structure comparison and structure similarity searching. *Nucleic acids research*, *32*(suppl_2), W582–W585.

Yi, C. et Huang, J. (2017). Semismooth newton coordinate descent algorithm for elastic-net penalized huber loss regression and quantile regression. *Journal of Computational and Graphical Statistics*, *26*(3), 547–557.

Zhang, B. et Horvath, S. (2005). A general framework for weighted gene co-expression network analysis. *Statistical applications in genetics and molecular biology*, *4*(1).

Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, *38*(2), 894–942.

Zhang, X., Wu, Y., Wang, L. et Li, R. (2016). Variable selection for support vector machines in moderately high dimensions. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, *78*(1), 53–76.

Zhu, J., Rosset, S., Tibshirani, R. et Hastie, T. (2003). 1-norm support vector machines. *Advances in neural information processing systems*, *16*.

Zou, H. et Hastie, T. (2003). Regression shrinkage and selection via the elastic net, with applications to microarrays. *JR Stat Soc Ser B*, *67*, 301–20.

Zou, H. et Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *The Annals of statistics*, *36*(4), 1509–1533.