Fast feature selection via knockoffs
Efficiently control the false discovery rate

School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne
Master thesis of

Quentin Rebjock

Supervised by

Prof. Alexandre d'Aspremont, Sierra team, Inria, ENS, CNRS
Prof. Martin Jaggi, Machine Learning and Optimization lab, EPFL

Paris, Inria, March 2020

Abstract

Feature selection is an active area of research in machine learning whose primary goal is to make models more interpretable by keeping most relevant covariates. Often, one is interested in selecting as many true positive features as possible (that is, maximizing true discoveries), while maintaining the false discovery rate under some threshold. In the past decades In this high-dimensional setting, that is when the number of features is larger than the number of samples, the problem of feature selection becomes particularly challenging.
Recently, Barber-Candès introduced the knockoffs framework to control the false discovery rate when performing feature selection. The key idea is to build a knockoff feature for each original feature
In this master thesis, we focus on the high-dimensional feature selection setting. Several bottleneck of the knockoff procedure are addressed.
Keywords: Feature selection, false discovery rate control, knockoff filter, high dimension

Acknowledgements

I would like to thank my master thesis advisor Alexandre d'Aspremont who gave me the opportunity to work. I would also like to thanks Armin Askari and Laurent El Ghaoui with whom I had the opportunity to work, and who provided substantial help for the resolution of SDP problems. Finally, I would like to thank my EPFL advisor Martin Jaggi.

Contents

Notations

- $\mathbb{N}$ denotes the set of natural integers and $\mathbb{R}$ the field of real numbers.
- Vectors are written in bold letters, e.g. $\mathbf{x}$, $\mathbf{y} \in \mathbb{R}^p$.
- $\mathbf{0}$ and $\mathbf{1}$ are the vectors whose all entries are 0 and 1 respectively, and whose sizes are inferred by the context.
- For a vector $\mathbf{v}$, we note $\|\mathbf{v}\|_0$ the number of non-zero entries of $\mathbf{v}$ (that is, its cardinality).
- Matrices are written in capital letters, e.g. $X \in \mathbb{R}^{n \times p}$.
- We may cast any vector $\mathbf{x} \in \mathbb{R}^p$ to a column matrix $x \in \mathbb{R}^{p \times 1}$ with the same entry values.
- We note $\odot$ the Hadamard (element-wise) product between two vectors or matrices.
- Given two matrices A and B, we note $\left[A, B\right]$ and $\left[A; B\right]$ their horizontal and vertical concatenation respectively (if dimensions match).
- The diag operator may be used in two contexts; either to transform a vector into a diagonal matrix whose diagonal entries match the vector, or to extract the diagonal of a matrix. For example,

$$\operatorname{diag} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \qquad \operatorname{diag} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

- The sets of $n \times n$ symmetric matrices, symmetric positive semidefinite, and symmetric positive definite are denoted by S^n, S^n_+ and S^n_{++} respectively.
- Functions $\mathbb{R} \to \mathbb{R}$ applied to vectors or matrices are implicitly performed element-wise, unless otherwise specified.
- $\mathbb{E}\left[\mathcal{X}\right]$ denotes the expectation of a random variable (or vector) $\mathcal{X}$.
- Given two random variables (or vectors) $\mathcal{X}$, $\mathcal{Y}$, we note $\mathcal{X} \perp\!\!\!\perp \mathcal{Y}$ the fact that they are independent.
- Given a set $\mathcal{S}$, $|\mathcal{S}|$ and $\#\mathcal{S}$ denote its cardinality (number of elements).

Introduction

Feature selection In the past two decades, acquiring data has become Datasets end up with a number of features of several dozens of thousands. Often, it is also easier to measure features than to label the sample with the right category. Indeed, the latter often requires a human and is very costly. in this setup, feature selection is particularly challenging as the number of observed samples might be much smaller than the number of covariates. It is for example the case of fMRI data and genomic data.

In this master thesis, The first chapter introduces the concept of feature selection, false discovery rate, and the knockoff framework developed by Barber-Candès. Chapter 2 introduces a sparse version of naive Bayes that scales linearly in the number of features against which it is trained.

Chapter 1

Feature selection

We introduce in this chapter the concepts of feature selection and false discovery rate control, as well as a short review of a few methods that are usually employed, including the Lasso and Benjamini–Hochberg-Yekutieli procedures.

1.1 Background on feature selection

1.1.1 Definitions

Feature selection is an active area of research in statistics and machine learning. It primarily consists in identifying the most relevant features (or covariates) explaining an observed variable. It is closely related to Occam's razor which is a principle stating that the simplest explanation is often the best one. More formally, let $\mathcal{X}$ be a p-dimensional random vector representing observed features and $\mathcal{Y}$ a random target that may depend on $\mathcal{X}$. For example, $\mathcal{X}$ could be the levels of expression of the genes of an individual, and $\mathcal{Y} \in \{0, 1\}$ a binary response indicating whether or not the person has some disease. We wish to find the subset of covariates $\mathcal{S} \subseteq \{1, \dots, p\}$ that best explains the target $\mathcal{Y}$, be it a regression or a classification problem. To do so, suppose that the joint vector $(\mathcal{X}, \mathcal{Y})$ follows some distribution, that is $(\mathcal{X}, \mathcal{Y}) \sim \mathcal{P}_{\mathcal{X},\mathcal{Y}}$. Even though finding the conditional distribution $\mathcal{P}_{\mathcal{Y}|\mathcal{X}}$ is beyond hope, one may be interested in finding on which subset $\mathcal{S}$ of features $\mathcal{P}_{\mathcal{Y}|\mathcal{X}}$ depends. Such a subset is not necessarily unique Markov blanket [**?**], b [**?**]. We define the set of null features $\mathcal{H}_0 \subseteq \{1, \dots, p\}$ as follows: $j \in \mathcal{H}_0$ if and only if $\mathcal{Y} \perp\!\!\!\perp \mathcal{X}_j \mid \mathcal{X}_{-j}$ (where $\mathcal{X}_{-j}$ denotes that all vector entries are kept except the jth). The goal is thus to find a procedure selection a subset of features $\hat{\mathcal{S}} \subseteq \{1, \dots, p\}$.

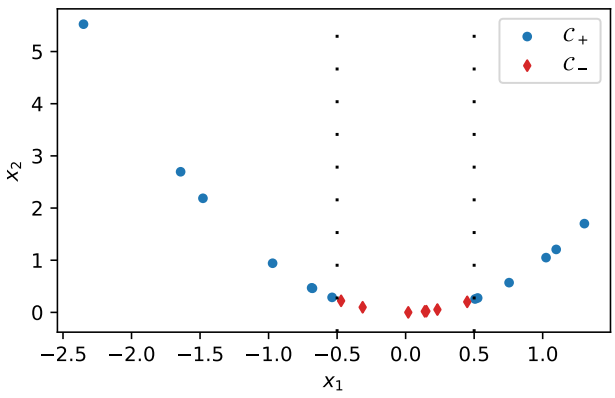

FIGURE 1.1: Illustration of the non-uniqueness of the Markov blanket. Here $\mathcal{X}_1 \sim \mathcal{N}(0, 1)$, $\mathcal{X}_2 = \mathcal{X}_1^2$, and $\mathcal{Y}$ is $\mathcal{C}_+$ if $|\mathcal{X}_1| \geq \frac{1}{2}$, $\mathcal{C}_-$ otherwise. Both $\mathcal{X}_1$ and $\mathcal{X}_2$ alone are enough to predict $\mathcal{Y}$.

In practice, one would observe several independent realizations of $(\mathcal{X}, \mathcal{Y})$ and would aggregate them into a feature matrix $X \in \mathbb{R}^{n \times p}$ of n samples and p features, and a target vector $\mathbf{y} \in \mathbb{R}^n$ respectively. The independence of the observations is a credible assumption in many real life settings

1.1.2 Motivation

Feature selection may be used in a multitude of contexts and we point out here the main reasons why one would want to perform it on a dataset.

1. Making the model more interpretable. It gives insights on the most relevant features to explain the observed target.

2. Facilitating data visualization. As only a small subset of the features is selected, projecting it to a low (2 or 3) dimensional space is easier.

3. Reducing the training time. Many machine learning algorithm have a super-linear time complexity in the number of features. Pre-selecting

a small subsets of those with a cheaper method can noticeably as less[?]

4. Improving the generalization of machine learning models. Irrelevant features may only be noise and fitting a model against them at train time is prone to over-fitting noise.

5. Avoiding the curse of dimensionality [?]. For example, the k-nearest neighbors algorithms [?] is known to perform badly as the feature space dimension increases. The number of samples actually has to grow exponentially in the number of features in order for the algorithm to perform decently.

Recently, the cost of measuring more features has drastically decreased. Many datasets, and especially in biology, end up with several dozens of thousands of features. Most of these features are expected to be insignificant, but there is a priori no reason to eliminate them. Adding them to the feature matrix is cheap, and it is then the role of the machine learning algorithm to detect the relevant ones.

1.1.3 Techniques

A lot of different paradigms and techniques to perform feature selection exist [?] a [?]. b[?] c[?] d[?]

Lasso

First rediscovered by Tibshirani [?] in 1996, the Lasso has become increasingly popular because of its capacity to both shrink the coefficients towards 0 and to select a subset of the features. It is basically a linear least squares regression whose weights are penalized by their ℓ_1 norm, multiplied by some factor $\lambda > 0$.

$$\hat{\boldsymbol{\beta}}(\lambda) = \arg\min_{\mathbf{b}} \frac{1}{2} \|\mathbf{y} - X\mathbf{b}\|_2^2 + \lambda \|\mathbf{b}\|_1 \tag{1.1}$$

The ℓ_1 penalty tends to make the weights $\hat{\boldsymbol{\beta}}(\lambda)$ sparse as λ increases. Actually, for any feature j, there is a $\lambda_{\min}$ such that for all $\lambda \geq \lambda_{\min}$, $\hat{\boldsymbol{\beta}}_j(\lambda) = 0$. That wouldn't be the case with an ℓ_2 penalty, for which the coefficients tend to 0 without reaching that value. This property makes the Lasso particularly suited for feature selection; just keep the features whose weight is non-zero. However, the choice of λ seems to be arbitrary, especially as it's not possible to know beforehand how many features will be selected.

The behavior of the path $\lambda \mapsto \hat{\boldsymbol{\beta}}(\lambda)$ has been extensively studied, and the algorithm LARS [?] was developed to efficiently compute it, which is possible because it is piecewise linear. It allows to compute $\hat{\boldsymbol{\beta}}$ for all relevant value of λ at a marginal cost. In practice, the λ giving the highest score on a *train* dataset is picked. The ℓ_1 regularization can easily be extended to many other estimators than the least squares, as for example logistic regression or SVMs.

Sparse center classifiers

Nearest centroid classification [?] is a very simple classification scheme. It consists in computing the averages $\boldsymbol{\theta}^{\pm} = \sum_{j \in \mathcal{I}^{\pm}} \mathbf{x}_i \in \mathbb{R}^p$ of the data points from the positive and negative classes $\mathcal{C}^{\pm}$, where $\mathcal{I}^{\pm}$ contain the positive and the negative points. A new data point $\mathbf{x} \in \mathbb{R}^p$ is classified positive or negative depending on its closest average $\boldsymbol{\theta}^+$ or $\boldsymbol{\theta}^-$. Finding the class averages can be formulated as the following optimization problem.

$$\arg\min_{\boldsymbol{\theta}^+, \boldsymbol{\theta}^-} \frac{1}{n^+} \sum_{i \in \mathcal{I}^+} \left\|\mathbf{x}_i - \boldsymbol{\theta}^+\right\|_2^2 + \frac{1}{n^-} \sum_{i \in \mathcal{I}^-} \left\|\mathbf{x}_i - \boldsymbol{\theta}^-\right\|_2^2 \tag{1.2}$$

Note Δ the decision boundary, such that $\mathbf{x}$ is classified $\mathcal{C}^+$ if $\Delta(\mathbf{x}) > 0$ and $\mathcal{C}^-$ otherwise.

$$\Delta(\mathbf{x}) = \left\|\mathbf{x} - \boldsymbol{\theta}^-\right\|_2^2 - \left\|\mathbf{x} - \boldsymbol{\theta}^+\right\|_2^2 \tag{1.3}$$

The expression 1.3 can be expanded into $\Delta(\mathbf{x}) = \|\boldsymbol{\theta}^+\|_2^2 + \|\boldsymbol{\theta}^-\|_2^2 + 2\mathbf{x}^\top(\boldsymbol{\theta}^+ - \boldsymbol{\theta}^-)$, making it clear that the decision depends on the feature j if and only if $\boldsymbol{\theta}_j^+ \neq \boldsymbol{\theta}_j^-$. Using this observation,[?] introduced a ℓ_0 penalized

version of 1.2 that can be solved very efficiently in closed-form. The authors obtain accuracy scores on par with the ones of the Lasso on the datasets they experiment. This a sparse centroid classification allows in particular to perform feature selection by keeping the features j such that $\boldsymbol{\theta}_j^+ \neq \boldsymbol{\theta}_j^-$.

None of the selection scheme presented above offer strong guarantees regarding the number of false positives that are detected. The selected subset $\hat{\mathcal{S}}$ could potentially be disjoint to $\mathcal{S}$ if the selection criterion is not adapted to the problem.

1.2 False discovery rate control

When performing feature selection one is usually interested in two quantities, namely the *false discovery rate* (FDR) and the *power*. Intuitively, the former (resp. the latter) assesses the expected proportion of false discoveries (resp. true discoveries) of a selection procedure.

1.2.1 Definitions

Suppose the conditional probability distribution $\mathcal{Y} \mid \mathcal{X}$ depends only on a subset of features $\mathcal{S} \subset \{1, \ldots, p\}$. A feature selection algorithm outputs a subset $\hat{\mathcal{S}} \subset \{1, \ldots, p\}$ (which is potentially random) that it judges to be relevant. The false discovery proportion (FDR), the false discovery rate (FDR), and the power are defined as follows

$$\mathrm{FDP} = \frac{\left|\{\hat{\mathcal{S}} \setminus \mathcal{S}\}\right|}{|\hat{\mathcal{S}}|}, \qquad \mathrm{FDR} = \mathbb{E}\left[\mathrm{FDP}\right], \qquad \mathrm{power} = \mathbb{E}\,\frac{\left|\{\hat{\mathcal{S}} \cap \mathcal{S}\}\right|}{|\hat{\mathcal{S}}|} \tag{1.4}$$

The FDP merely measures the proportion of features in $\hat{\mathcal{S}}$ that are not in $\mathcal{S}$. It depends on the samples X and $\mathbf{y}$, and possibly on the inherent randomness of the selection algorithm. That is why the FDR assesses the expectation of this value. Finally, the power is the fraction of the important features that were actually discovered, in average.

Even though it is beyond hope to retrieve the whole set $\mathcal{S}$ with no error, a multitude of techniques attempt to find as many relevant features as possible (that is, maximizing the power) while maintaining the FDR under a certain threshold. The concept of FDR was first introduced by [?] in 1995 and has become increasingly decisive in some sciences such has biology where the acquisition of a huge number of covariates is frequent. It is now possible to measure the expression of several dozens of thousands of genes for a given individual.

1.2.2 Benjamini–Hochberg-Yekutieli procedures

The Benjamini–Hochberg [?] and Benjamini–Yekutieli [?] schemes are two methods controlling the FDR that are widely used in practice. They are closely related to each other but offer different guarantees regarding the effective control of the FDR. BH is less conservative than BY but makes stronger assumptions on the p-values it manipulates.

For each feature $j \in \{1, \ldots, p\}$, let $\mathcal{H}_j$ be the null hypothesis (j does not belong to cS), and p_j the corresponding p-value. Let $q \in [0, 1]$ be some FDR target that should not be exceeded. We note $(p_{(j)})_j$ the sequence of p-values in increasing order; $p_{(1)} \leq \cdots \leq p_{(p)}$. Let m_0 be the number of true null hypotheses.

Benjamini–Hochberg

The Benjamini–Hochberg (BH) procedure was the first method proposed to control the FDR. It consists in the following steps:

1. Sorting the p-values such that $p_{(1)} \leq \ldots \leq p_{(p)}$
2. Finding the largest $k \in \mathbb{N}$ such that $p_{(k)} \leq \frac{k \cdot q}{p}$
3. Rejecting all the hypotheses $\mathcal{H}_{(j)}$ such that $j \in \{1, \ldots, k\}$

In the end, all the features j such that $p_j \leq p_{(k)}$ are selected. It guarantees that FDR $\leq \frac{m_0}{p} q$ under the assumption that the p-values were computed independently, which can be very restrictive in many situations.

Benjamini–Yekutieli

Similarly, the Benjamini–Yekutieli procedure controls the FDR and does not need the p-values independence assumption, at the cost of a more conservative selection. It follows the same scheme, namely: ordering the p-values, finding the largest $k \in \mathbb{N}$ such that $p_{(k)} \leq \frac{k}{p \cdot c(p)} q$, and rejecting $\mathcal{H}_{(j)}$ if $j \leq k$. Note the additional factor $c(p) \geq 1$, defined in 1.5.

$$c(p) = \sum_{j=1}^{p} \frac{1}{j}, \qquad \text{FDR} \leq \frac{m_0}{p} q \tag{1.5}$$

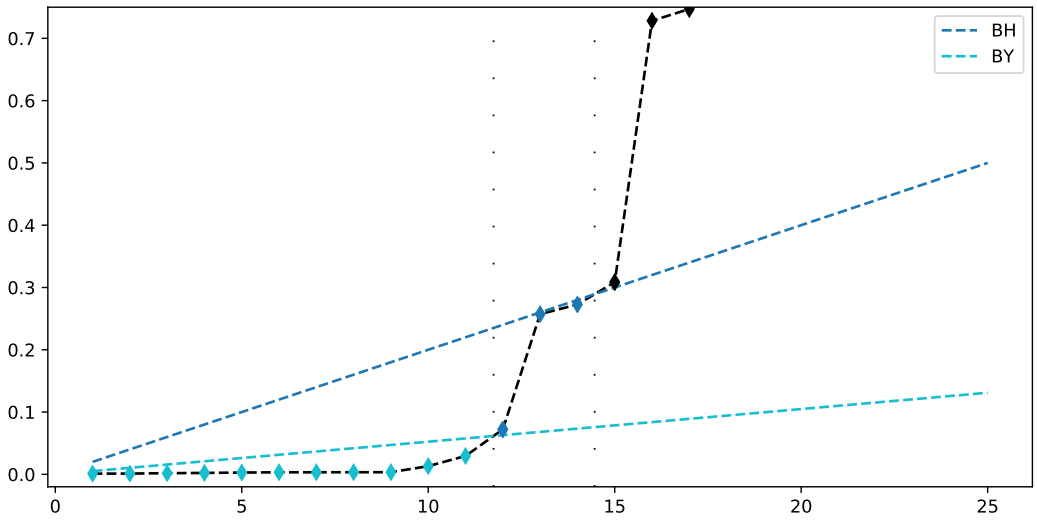

FIGURE 1.2: Illustration of the BH and the BY procedures. They are geometrically equivalent to drawing a line of slope β going through the origin, identifying the last p-value under the line, and keeping the features on the left of the intersection point. For BH, $\beta = \frac{q}{p}$ (in blue), while for BY, $\beta = \frac{q}{p \cdot c(p)}$ (in cyan). In this toy example, $q = 0.5$, and it shows that BY is way more conservative than BH.

Both procedures are illustrated in figure 1.2. They require p-values to be computed, which is not always feasible, especially when $p > n$. Furthermore, BH and BY have stronger guarantees than desired; for a desired FDR level q, they ensure that FDR $\leq \frac{m_0}{p} q$, and the factor $\frac{m_0}{p}$ might be much smaller than 1.

Chapter 2

The knockoffs filter framework

In 2015 Barber-Candès introduced the knockoffs framework and extended it later in 2018 to a more general setting (allowing in particular to perform feature selection in the high dimension context). The goal is to control the false discovery rate as defined in 1.4, while maintaining a reasonable power. More formally, let $X \in \mathbb{R}^{n \times p}$ be a a feature matrix and $\mathbf{y} \in \mathbb{R}^n$ the associated target vector. Suppose $\mathbf{y}$ depends only on the subset of features $\mathcal{S} \subseteq \{1, \dots, p\}$. Given some FDR target $q \in [0, 1]$, we wish to find a procedure such as, in average, the false discovery proportion is smaller that q, i.e. $\text{FDR} \leq q$. To do so, the key idea is to construct for each original feature X_j, $j \in \{1, \dots, p\}$, a knockoff (that is to say, fake) feature $\tilde{X}_j$ which is known to be out of the model. Original features are then selected only if they prove to be more significant than their knockoff counterparts.

Note $\Sigma = X^\top X$.

To compare a feature and its knockoff several quantities will be computed by interchanging them. For this reason, we define in 1 the swap operator.

Definition 1. *(*swap *operator) Given two matrices A, $B \in \mathbb{R}^{n \times p}$ of same size, we define the* swap *operator on the concatenated matrix $[A, B]$ as follows. For a any subset of indices $S \subseteq \{1, \dots, p\}$, $[A, B]_{\text{swap}(S)}$ is the transformed matrix where the columns A_j and B_j were swapped for all $j \in S$.*

In the following sections, we are going to detail principal aspects of the construction of the knockoff features, the computation of statistics for both original and knockoff features, and the feature selection itself, based on those statistics.

2.1 Knockoffs construction

In 2015, Barber-Candès introduced first the *fixed*-X knockoffs variable selection procedure. It relies on the creation of fake features satisfying some correlation constraints with the original features. Unfortunately, it can only perform adequately when $n \geq 2p$, even though it can be partially extended to the cases where $n \geq p$. In 2018, Candès proposed an extension of the framework called *model*-X knockoffs, in which fake features are sampled from a learned distribution. Despite the restriction of *fixed*-X knockoffs, that method is still appealing as the construction of the knockoff variables is straightforward (but costly). On the other hand, *model*-X knockoffs work in higher feature dimensions but need an estimation of the distribution that generated the data, which is a hard problem in general. We will focus on the construction of *model*-X knockoffs as we are principally interested in the high dimensional setting.

In this section, let $(\mathcal{X}, \mathcal{Y})$ be a pair of random variables, $\mathcal{X}$ being composed of p features, and $\mathcal{Y}$ the associated target in $\mathbb{R}$. We suppose that the feature matrix and the target vector $(X, \mathbf{y})$ are composed of samples from $(\mathcal{X}, \mathcal{Y})$. We wish to build knockoff features $\tilde{X} \in \mathbb{R}^{n \times p}$, and to do so we are going to sample from a random variable $\tilde{\mathcal{X}}$ built such that $[\mathcal{X}; \tilde{\mathcal{X}}]$ follows the properties S.1 and S.2 defined thereafter.

2.1.1 General case

Definition 2. *(model-X knockoffs) Given random vector $\mathcal{X} \in \mathbb{R}^p$ of features, a random vector $\tilde{\mathcal{X}} \in \mathbb{R}^p$ is said to be model-$\mathcal{X}$ knockoffs with respect to $\mathcal{Y}$ if it satisfies the two following properties:*

S.1 *For any $S \subseteq \{1, \dots, p\}$, $[\mathcal{X}; \tilde{\mathcal{X}}]^\top_{\text{swap}(S)} \overset{d}{=} [\mathcal{X}; \tilde{\mathcal{X}}]^\top$*

S.2 $\tilde{\mathcal{X}} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X}$

Intuitively, the condition S.1 ensures that a knockoff feature is sufficiently close to its associated original feature so that swapping them doesn't change the distribution of the concatenated random vector. The independence condition S.2 states that the knockoff features carry no additional information on $\mathcal{Y}$, given $\mathcal{X}$. It is trivially satisfied if $\tilde{X}$ is built without exploiting $\mathbf{y}$. However, constructing knockoffs meeting the first distribution equality is practically infeasible in general.

Remark 1. *Constructing the knockoffs feature matrix $\tilde{X} \in \mathbb{R}^{n \times p}$. It must satisfy two conditions, namely:*

$$\tilde{X}^\top \tilde{X} = \Sigma,$$
$$X^\top \tilde{X} = \Sigma - \operatorname{diag}\{\mathbf{s}\} \text{ where } \mathbf{s} \text{ is a non negative vector.}$$

It ensures that $\tilde{X}$ has the same covariance as the original matrix X and that the correlation between distinct original and knockoff variables is the same as the correlation between the two originals.

2.1.2 Gaussian case

In the particular case where $\mathcal{X}$ is multivariate gaussian, the exact distribution of $\tilde{\mathcal{X}}$ can be derived.

Proposition 2. *Suppose that $\mathcal{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. If $\tilde{\mathcal{X}}$ is a random variable such that*

$$[\mathcal{X}; \tilde{\mathcal{X}}] \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu} \end{bmatrix}, \Omega \right), \quad \text{where} \quad \Omega = \begin{bmatrix} \Sigma & \Sigma - \operatorname{diag} \mathbf{s} \\ \Sigma - \operatorname{diag} \mathbf{s} & \Sigma \end{bmatrix} \quad \text{for some } \mathbf{s} \in \mathbb{R}^p,$$

then $[\mathcal{X}, \tilde{\mathcal{X}}]$ satisfies the swap *property S.1, provided that Ω is positive semidefinite (so that it is indeed a covariance matrix).*

In Proposition 2, $\mathbf{s} \in \mathbb{R}^p$ can be any vector such that $\Omega \succeq \mathbf{0}$. We will come back later to the choice of $\mathbf{s}$ which is actually crucial. For now, assume that $\mathbf{s}$ satisfies this assumption. This result gives a way of constructing the knockoff features from X. Indeed, as $[\mathcal{X}; \tilde{\mathcal{X}}]$ is multivariate normal, we may compute the exact distribution of $\tilde{\mathcal{X}} \mid \mathcal{X}$ with classical formulas [**?**] as shown in 5.1.

$$\tilde{\mathcal{X}} \mid \mathcal{X} \sim \mathcal{N}(\boldsymbol{v}, \Upsilon), \qquad \text{where} \quad \begin{cases} \boldsymbol{v} = \mathcal{X} - \mathcal{X}\Sigma^{-1} \operatorname{diag}(\mathbf{s}) \\ \Upsilon = \operatorname{diag}(\mathbf{s}) \left(2I_{p \times p} - \Sigma^{-1} \operatorname{diag}(\mathbf{s})\right) \end{cases} \tag{2.1}$$

To put this into practice, one would compute the empirical mean $\hat{\boldsymbol{\mu}} \in \mathbb{R}^p$ and covariance $\hat{\Sigma} \in \mathbb{R}^{p \times p}$ of $\mathcal{X}$ using the observed feature matrix X. Then, each row of $\tilde{X}$ is sampled according to a gaussian distribution $\mathcal{N}(\boldsymbol{v}, \Upsilon)$ whose parameters are described in 5.1. Note in particular that the construction process is random; if it is repeated, it may very well return different knockoffs, and thus different selected features. Because of this instability, several attempts [**?**] to fix it by aggregating several knockoff samples.

Depending on the prior on the data and on the available computing power, several algorithms may be used to estimate the covariance matrix Σ. The empirical estimator is cheap but known to be unstable when $p > n$. Shrunk estimations like Ledoit-Wolf [**?**] provide good results.

The gaussian hypothesis is obviously rarely verified in practice but yields acceptable results even when $\mathcal{X}$ is far from gaussian. It partly comes from the fact that, rather than constructing $\tilde{\mathcal{X}}$ to respect S.1, a weaker condition would be to enforce $[\mathcal{X}, \tilde{\mathcal{X}}]$ and $[\mathcal{X}, \tilde{\mathcal{X}}]_{\text{swap}(S)}$ to have the same first two moments (mean and covariance). It turns out to be the case if $\tilde{\mathcal{X}} \mid \mathcal{X}$ is constructed as described in 5.1. In the remaining of this master thesis, we will restrain ourselves to the gaussian hypothesis.

The first knockoff paper [**?**] introducing the *fixed*-X knockoffs relied on similar correlation properties between the original and the knockoff features. It however imposed $p \leq n$ and fewer statistics would yield theoretical guarantees regarding the FDR control of the procedure. In the reminding, we will only consider *model*-X knockoffs as we are interested in the high dimensional setting. Moreover, *model*-X knockoffs tend to give higher power experimentally compared to their *fixed* counterparts.

Note that generating knockoffs requires an estimation of the distribution of $\mathcal{X}$ only, and not $\mathcal{Y} \mid \mathcal{X}$ as most methods would. It is particularly appealing because labeling data is often the most costly part, while acquiring samples $\mathbf{x} \sim \mathcal{X}$ is easier. Unlabeled samples are thus valuable.

2.2 Statistics computation

2.2.1 General principle

Given the original feature matrix and the sampled knockoffs $X, \tilde{X} \in \mathbb{R}^{n \times p}$, statistics w_j for all $j \in \{1, \dots, p\}$ are computed. Each w_j represents how more significant the original feature X_j is compared to $\tilde{X}_j$. These statistics must satisfy the *flip-sign* technical condition 3 for the FDR control to carry out, but a wide variety of choices is possible as will be shown.

Definition 3. *(flip-sign property) A statistics function $\omega\colon \mathbb{R}^{n \times 2p} \times \mathbb{R}^n \to \mathbb{R}^p$ is said to follow the flip-sign property if for any $S \subseteq \{1, \dots, p\}$ and any $j \in \{1, \dots, p\}$,*

$$\omega_j\big([X, \tilde{X}]_{\text{swap}(S)}, \mathbf{y}\big) = \begin{cases} -\omega_j\big([X, \tilde{X}], \mathbf{y}\big) & \text{if } j \in S. \\ \omega_j\big([X, \tilde{X}], \mathbf{y}\big) & \text{otherwise.} \end{cases}$$

This property is very natural, as it is simply asking the original and the knockoff features to play antisymmetric roles.

2.2.2 Statistics aggregation

Constructing statistics satisfying the *flip-sign* property 3 is actually straightforward as an elementary scheme in two steps leads to such statistics. The idea is to build statistics for each original and each knockoff feature, and then aggregate them.

1. First construct statistics $[\mathbf{z}; \tilde{\mathbf{z}}] = \zeta\big([X, \tilde{X}], \mathbf{y}\big)$ with some function $\zeta\colon \mathbb{R}^{n \times 2p} \times \mathbb{R}^n \to \mathbb{R}^{2p}$ satisfying

$$[\mathbf{z}; \tilde{\mathbf{z}}]_{\text{swap}(S)} = \zeta\big([X, \tilde{X}]_{\text{swap}(S)}, \mathbf{y}\big), \quad \forall S \subseteq \{1, \dots, p\} \tag{2.2}$$

   The statistics z_j (resp. $\tilde{z}_j$) indicates how significant the original (resp. knockoff) feature j is.

2. Then aggregate for each $j \in \{1, \dots, p\}$ the statistics of the original feature z_j and the one of the corresponding knockoff $\tilde{z}_j$ with an antisymmetric function $a_j\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. That is, set $w_j = a_j(z_j, \tilde{z}_j)$.

It is easy to show that such constructed statistics will satisfy the *flip-sign* property 3.

Basically any antisymmetric function a_j could work, but some choices experimentally lead to a better power. Here are a few examples of mappings:

- $w_j = z_j - \tilde{z}_j$ (experimentally gives highest power in many cases)
- $w_j = \max(z_j, \tilde{z}_j) \times \operatorname{sign}(z_j - \tilde{z}_j)$ (first proposed in 2015)
- $w_j = \log \frac{z_j}{\tilde{z}_j}$

As for the function ζ, it only needs to satisfy the *swap* property 2.2. This condition may seem restrictive but a large number of choices are actually valid.

2.2.3 Examples

A few examples of statistics relying on the aggregation trick are presented in this subsection.

In [**?**], Barber-Candès suggest after empirical observations the use of the Lasso Signed Max (LSM) statistics defined as follows.

$$z_j = \sup\{\lambda \mid \hat{\boldsymbol{\beta}}_j(\lambda) \neq 0\}, \qquad \hat{\boldsymbol{\beta}}(\lambda) = \arg\min_{\mathbf{b}} \frac{1}{2}\big\|\mathbf{y} - [X, \tilde{X}]\mathbf{b}\big\|_2^2 + \lambda \|\mathbf{b}\|_1 \tag{2.3}$$

The vector $\hat{\boldsymbol{\beta}}(\lambda) \in \mathbb{R}^{2p}$ contains the coefficients of a Lasso model with penalty coefficient $\lambda > 0$. As mentioned in subsection 1.1.3, all the coefficients are null starting from $\lambda = +\infty$, and are likely to shift (and to grow in absolute value) as $\lambda \to 0$. It makes sense to use the first point where the coefficient β_j is not null as a significance metric for each individual feature j. In addition, the LARS algorithms is able to compute that value pretty efficiently [**?**].

Training a plain Lasso estimator. Once again, the LARS algorithm allows to cross-validate the Lasso efficiently. Experimentally, the Lasso gives remarkable results in terms of power.

More generally, the coefficients in absolute value of any reasonable regressor (or classifier, depending on the task) constitute a judicious option of statistics. The fact that the *flip-sign* property isn't restrictive makes the knockoff framework very powerful. Depending on the data, an adapted model can be trained be it logistic regression, SVMs, or random forests [**?**]. When the model has hyper-parameters, those are typically tuned using cross-validation. This step is usually costly.

2.3 Selection thresholds

The selection itself requires the computation of a data-dependent threshold τ conditioned by the target FDR $q \in [0, 1]$.Finally, only the features j whose statistics w_j is above the threshold are selected.

$$\hat{\mathcal{S}} = \{j \mid w_j \geq \tau\} \tag{2.4}$$

Depending on how restrictive the procedure ought to be, the threshold τ can be adapted. It leads to different guarantees regarding the control of the FDR. Barber-Candès establish two selection procedures, namely *knockoff* and *knockoff+*. They control the modified FDR (as defined below) and the FDR respectively.

Definition 4. *Given an estimate $\hat{\mathcal{S}}$ of $\mathcal{S}$ and a desired target false discovery rate $q \in [0, 1]$, we define the modified FDR as follows:*

$$mFDR = \mathbb{E}\,\frac{\left|\left\{j \mid j \in \hat{\mathcal{S}} \setminus \mathcal{S}\right\}\right|}{|\hat{\mathcal{S}}| + 1/q}$$

As $0 \leq q \leq 1$, mFDR as defined in 4 is always smaller than the actual FDR. It means that controlling the mFDR is less restrictive than controlling the FDR, as the FDR could potentially be arbitrarily larger than the mFDR. But if the target threshold q is not too small, and if many features are selected by the procedure, the modified version of the FDR is close enough to the actual FDR. As will be shown later, being less restrictive by controlling only the mFDR can greatly improve the power of the selection.

The only difference between the *knockoff* and the *knockoff+* procedures is the selection threshold τ.

Definition 5. *(knockoff and knockoff+ thresholds) Given the statistics $\mathbf{w} \in \mathbb{R}^p$ computed from X and $\tilde{X}$, we define the knockoff and knockoff+ threshold respectively as follows:*

$$\tau = \min\left\{t > 0 \mid \frac{|\{j \mid w_j \leq -t\}|}{|\{j \mid w_j \geq t\}|} \leq q\right\} \tag{2.5}$$

$$\tau^+ = \min\left\{t > 0 \mid \frac{1 + |\{j \mid w_j \leq -t\}|}{|\{j \mid w_j \geq t\}|} \leq q\right\} \tag{2.6}$$

They only differ by their numerator, where τ^+ has an additional $+1$. The main result of Barber-Candès regarding the control of the mFDR and FDR is stated in 3.

Theorem 3. *(guarantees of the knockoff procedures) Construct $\hat{\mathcal{S}} = \{j \mid w_j \geq \tau\}$ and $\hat{\mathcal{S}}^+ = \{j \mid w_j \geq \tau^+\}$. These selections ensure the following FDR controls:*

$$mFDR\left[\hat{\mathcal{S}}\right] \leq q, \qquad FDR\left[\hat{\mathcal{S}}^+\right] \leq q \tag{2.7}$$

Even if only the *knockoff+* method truly control the FDR, using the threshold τ improves the power and gives reasonable FDR, in the same way the BH procedure usually controls the FDR even when the tests are not independent.

2.4 Bottlenecks

Despite the nice theoretical guarantees on the FDR control that the knockoff procedure proposes, two bottlenecks hurt its performances in the high dimensional setting.

2.4.1 SDP

In the gaussian setting, knockoff features are sampled according to the distribution 5.1. The control guarantees hold for any $\mathbf{s} \in \mathbb{R}^p$ such that the covariance matrix Ω is semidefinite positive.

Proposition 4. *Let* $\Omega = \begin{bmatrix} \Sigma & \Sigma - \operatorname{diag} \mathbf{s} \\ \Sigma - \operatorname{diag} \mathbf{s} & \Sigma \end{bmatrix}$. *Then* $\Omega \succeq \mathbf{0}_{p \times p}$ *if and only if* $2\Sigma \succeq \operatorname{diag} \mathbf{s} \succeq \mathbf{0}$.

Proof. Note $D = \operatorname{diag} \mathbf{s}$. By computing the Schur complement [?] $\Omega_{/\Sigma} = 2D - D\Sigma^{-1}D$, $\Omega \succeq \mathbf{0}$ is equivalent to $\Omega_{/\Sigma} \succeq \mathbf{0}$. Define M and its Schur complements as follows

$$M = \begin{bmatrix} 2D & D \\ D & \Sigma \end{bmatrix}, \qquad \begin{cases} M_{/2D} = \Sigma - \frac{1}{2}D \\ M_{/\Sigma} = 2D - D\Sigma^{-1}D \end{cases}$$

Finally, $\Omega_{/\Sigma} = M_{/\Sigma}$ is p.s.d. if and only if both $\Sigma - \frac{1}{2}D$ and D are p.s.d. □

2.4.2 Statistics computation

and could take advantage of sparse naive Bayes to run faster.

These two bottlenecks make the . The following chapters tackle these two issues by proposing fast statistics computation. From now, only binary classification problems will be considered, where the label y takes values in $\{\mathcal{C}^+, \mathcal{C}^-\}$.

2.5 Python implementation

Barber-Candès (along with other coauthor) provided a R and MATLAB implementation of the knockoff framework[1]. To the best of our knowledge, no unified Python implementation The implementation follows deliberately the Scikit-Learn [?] structure. It provides 3 main components, namely a knockoffs generator `KnockoffsGenerator`, statistics computations utility functions, and a selector `BaseKnockoffsSelector`. `KnockoffsGenerator` and `BaseKnockoffsSelector` subclass `BaseEstimator` and `TransformerMixin` and `BaseEstimator` and `SelectorMixin` respectively.

Code 1 Example of knockoffs usage

```
selector = SimpleKnockoffsSelector(
    GaussianXKnockoffs(),
    z_to_w_statistics(estimator_statistics(estimator=LassoCV())),
    alpha=0.2,
)

selector.fit(X, y)
print(f'Selected features: {selector.mask_}')
```

[1] The R package page can be found at this address. Sources for both R and MATLAB are stored in this GitHub repository.

Chapter 3

Sparse naive Bayes

Naive Bayes was first introduced in the early 60s by [?] for text documents classification. It is a simple model assuming the independence of the features, given the label. Despite this naive presupposition, naive Bayes remains an engaging model for large scale datasets because of its low complexity. The time complexity to train a model is asymptotically $\mathcal{O}(n \cdot p)$, where n and p are the number of samples and the number of features respectively. Another appealing propriety is that naive Bayes can be trained in an online fashion, as new data points come in a sequential order. It can be particularly helpful if the dataset doesn't fit in memory. Furthermore, distributed implementations could even be considered in order to speed up the learning process.

In this chapter, classical naive Bayes along with a few notations are briefly introduced. Then, a sparse version of naive Bayes is presented, allowing in particular to perform feature selection.

3.1 Reminders on vanilla naive Bayes

In this section, we recall briefly the naive Bayes model in the specific case of binary classification, and some details regarding the training phase under the bernoulli, multinomial and gaussian underlying assumptions. Most key results can naturally be extended to the multiple classes setting.

Let n and p be two integers, $X \in \mathbb{R}^{n \times p}$ a design matrix and $\mathbf{y} \in \{-1, 1\}^n$ the associated target vector. The negative and the positive classes are noted $\mathcal{C}_-$ and $\mathcal{C}_+$ respectively, and will be indifferently substituted with their respective labels.

3.1.1 General settings

Note $(\mathcal{X}, \mathcal{Y})$ the pair of random variables that generated the samples and targets X and $\mathbf{y}$. The goal is to explain $\mathbf{y}$ given X, that is, finding the posterior probabilities $\Pr(\mathcal{C}_\pm \mid \mathcal{X})$. Given these probabilities, a new observation $\mathbf{x} \in \mathbb{R}^p$ is classified according to the highest posterior probability between the two classes

$$y(\mathbf{x}) = \underset{\mathcal{C} \in \{\mathcal{C}^-, \mathcal{C}^+\}}{\arg\max} \Pr(\mathcal{C} \mid \mathbf{x}) \tag{3.1}$$

To do so, we combine the use of Bayes rule and make the (very) naive assumption that all the features are independent given the class, that is

$$\Pr(\mathcal{X} = \mathbf{x} \mid \mathcal{C}_\pm) = \prod_{j=1}^{p} \Pr(\mathcal{X}_j = x_j \mid \mathcal{C}_\pm), \qquad \Pr(\mathcal{C}_\pm \mid \mathcal{X} = \mathbf{x}) = \frac{\Pr(\mathcal{X} = \mathbf{x} \mid \mathcal{C}_\pm) \cdot \Pr(\mathcal{C}_\pm)}{\Pr(\mathcal{X} = \mathbf{x})} \tag{3.2}$$

On the right hand side, the denominator $\Pr(\mathcal{X} = \mathbf{x})$ does not depend on the class $\mathcal{C}_\pm$. It is therefore not required to evaluate it in order to perform the inference described in 3.1. Thus, we only need to estimate the probabilities $\Pr(\mathcal{C}_\pm)$ and $\Pr(\mathbf{x} \mid \mathcal{C}_\pm)$. The former are simply data averages, i.e. the frequencies of the positive and negative classes in the observed data. As for the latter probabilities $\Pr(\mathbf{x} \mid \mathcal{C}_\pm) = \prod_{j=1}^{p} \Pr(x_j \mid \mathcal{C}_\pm)$, they can be modeled by a plethora of distribution families, depending on the prior knowledge we have on the data. The distribution is typically parametrized by some vector $\boldsymbol{\theta} \in \mathbb{R}^m$. The probabilities are usually computed by maximizing the likelihood $\mathcal{L}$, or equivalently the log-likelihood $\mathcal{LL} = \log \mathcal{L}$, of the observed data.

$$\mathcal{LL}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \Pr(\mathbf{x}_i \mid y_i ; \boldsymbol{\theta})$$

We present now 3 meaningful cases, where the prior distributions of $\Pr(\mathbf{x} \mid \mathcal{C}_\pm)$ are either gaussian, bernoulli or multinomial. Let $\mathcal{I}_\pm = \{i \in \{1, \dots, n\} \mid y_i = \mathcal{C}_\pm\}$ be the sets containing the indexes of the positive and negative data points. We also note for each class $\mathcal{C}_\pm$ their cardinalities and empirical sums respectively, as follows:

$$n^\pm = |\mathcal{I}_\pm|, \qquad \mathbf{f}^\pm = \sum_{i \in \mathcal{I}_\pm} \mathbf{x}_i$$

3.1.2 Gaussian naive Bayes

In this case the observed data conditioned on its label is modeled by the gaussian distribution $\mathcal{N}(\boldsymbol{\mu}^\pm, \Sigma^\pm)$. It is the most common configuration because it can account for continuous data and the normal distribution constitures a decent prior in many cases by virtue of its high entropy. Note that the covariances $\Sigma^\pm \in \mathbb{R}^{p \times p}$ are diagonal as a result of the independence assumption that we made.

$$\Pr(\mathbf{x} \mid C_\pm) = \frac{1}{\sqrt{(2\pi)^p \det(\Sigma_\pm)}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_\pm)^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_\pm)\right)$$

By denoting $\sigma_j = \Sigma_{jj}$, the log-likelihood can be written

$$\begin{aligned}\mathcal{LL}_g(\boldsymbol{\mu}_+, \boldsymbol{\sigma}_+, \boldsymbol{\mu}_-, \boldsymbol{\sigma}_-) = \sum_{j=1}^{p} \Bigg[&\sum_{i \in \mathcal{I}_+} -\frac{1}{2}\log(2\pi) - \log \sigma_j^+ - \frac{(x_j - \mu_j^+)^2}{2{\sigma_j^+}^2} \\ &+ \sum_{i \in \mathcal{I}_-} -\frac{1}{2}\log(2\pi) - \log \sigma_j^- - \frac{(x_j - \mu_j^-)^2}{2{\sigma_j^-}^2} \Bigg]\end{aligned}$$

Even if $\mathcal{LL}_g$ is not concave, it maximizer admits a closed-form solution (readily shown to be the point where the gradient is null)

$$\boldsymbol{\mu}^\pm = \frac{\mathbf{f}^\pm}{n^\pm}, \qquad \boldsymbol{\sigma}^\pm = \sqrt{\frac{1}{n^\pm} \sum_{i \in \mathcal{I}^\pm} (\mathbf{x}_i - \boldsymbol{\mu}^\pm)^2}$$

3.1.3 Bernoulli naive Bayes

The Bernoulli distribution assumes that the design matrix is binary, that is $X \in \{0, 1\}^{n \times p}$. Even though this situation isn't very prevalent in practice, it has a simple and elegant solution. In order to model the conditional probabilities, we may assume the existence of $\boldsymbol{\theta}^+, \boldsymbol{\theta}^- \in (0, 1)^p$ such that for any data point $\mathbf{x} \in \mathbb{R}^p$,

$$\Pr(x_j \mid \mathcal{C}_\pm) = (\theta_j^\pm)^{x_j} \cdot (1 - \theta_j^\pm)^{1 - x_j}$$

It yields that $\log \Pr(\mathbf{x} \mid \mathcal{C}_\pm) = \mathbf{x}^\top \log \boldsymbol{\theta}^\pm + (\mathbf{1} - \mathbf{x})^\top \log(\mathbf{1} - \boldsymbol{\theta}^\pm)$ and finally

$$\begin{aligned}\mathcal{LL}_b(\boldsymbol{\theta}^+, \boldsymbol{\theta}^-) &= \sum_{i \in \mathcal{I}^+} \log \Pr(\mathbf{x}_i \mid \mathcal{C}_+) + \sum_{i \in \mathcal{I}^-} \log \Pr(\mathbf{x}_i \mid \mathcal{C}_-) \\ &= \mathbf{f}_+^\top \log \boldsymbol{\theta}^+ + (n_+ \mathbf{1} - \mathbf{f}_+)^\top \log\left(\mathbf{1} - \boldsymbol{\theta}^+\right) \\ &\quad + \mathbf{f}_-^\top \log \boldsymbol{\theta}^- + (n_- \mathbf{1} - \mathbf{f}_-)^\top \log\left(\mathbf{1} - \boldsymbol{\theta}^-\right)\end{aligned} \tag{3.3}$$

The independence assumption makes the optimization problem decomposable across features; it reduces to p simpler maximizations, each of them being concave and admitting a closed-form solution. Finally, we find that

$$\theta_\pm^\star = \frac{f^\pm}{n^\pm}, \qquad \text{which are simply the averages of each class.}$$

3.1.4 Multinomial naive Bayes

Multinomial naive Bayes generalizes the Bernoulli version as it supposes that $X \in \mathbb{N}^{n \times p}$ is generated by the following underlying distribution

$$\Pr(\mathbf{x} \mid \mathcal{C}_\pm) = \frac{\left(\sum_{j=1}^p x_j\right)!}{\prod_{j=1}^p x_j!} \cdot \prod_{j=1}^p (\theta_j^\pm)^{x_j} \tag{3.4}$$

In the special case where $X \in \{0, 1\}^{n \times p}$ we recover the above Bernoulli formulation. It is parametrized by $\boldsymbol{\theta}^+, \boldsymbol{\theta}^- \in (0, 1)^p$, and they must satisfy $\mathbf{1}^\top \boldsymbol{\theta}^+ = \mathbf{1}^\top \boldsymbol{\theta}^- = 1$ for 3.4 to be a proper distribution. Note that this model is still valid in the more general case where we only assume the data to be non-negative, $X \in \mathbb{R}_+^{n \times p}$. Hence, it is not as restrictive as it may appear at first sight, as it is applicable to a large number of datasets. The log probability is given by

$$\log \Pr(\mathbf{x} \mid \mathcal{C}_\pm) = \mathbf{x}^\top \log \boldsymbol{\theta}_\pm + \log \frac{\left(\sum_{j=1}^p x_j\right)!}{\prod_{j=1}^p x_j!}$$

and the log-likelihood reduces to (after removing the constant terms)

$$\mathcal{LL}_m\left(\boldsymbol{\theta}^+, \boldsymbol{\theta}^-\right) = \mathbf{f}_+^\top \log \boldsymbol{\theta}^+ + \mathbf{f}_-^\top \log \boldsymbol{\theta}^- \tag{3.5}$$

which is again decomposable across features. It turns out that the optimums are

$$\boldsymbol{\theta}_\pm^\star = \frac{\mathbf{f}^\pm}{\mathbf{1}^\top \mathbf{f}^\pm}$$

In the models presented above, the time complexity to train the naive Bayes classifier is $\mathcal{O}(n \cdot p)$. With a larger number of classes besides $\mathcal{C}^-$ and $\mathcal{C}^+$, say k classes, this complexity is multiplied by k. On top of this low asymptotic complexity, the solutions can be computed in closed-form, which makes the effective computation cost very low. In comparison, no closed-form solution exist for the Lasso, for logistic regression [?], and for SVMs [?]. These models are usually trained by employing more costly gradient-descent based algorithms to find the global optimum.

3.1.5 Decision boundary

Given a new data point $\mathbf{x} \in \mathbb{R}^p$, we wish to attribute it the most probable label $y \in \{\mathcal{C}^-, \mathcal{C}^+\}$. No matter what model parametrized by $\boldsymbol{\theta}$ was chosen, 3.1 reduces to

$$\begin{aligned} y &= \operatorname*{arg\,max}_{\mathcal{C} \in \{\mathcal{C}^-, \mathcal{C}^+\}} \Pr(\mathcal{C} \mid \mathbf{x}) \\ &= \operatorname{sign} \log \frac{\Pr(\mathcal{C}^+ \mid \mathbf{x}\,; \boldsymbol{\theta})}{\Pr(\mathcal{C}^- \mid \mathbf{x}\,; \boldsymbol{\theta})} \\ &= \operatorname{sign}\left[\log \frac{\Pr(\mathcal{C}^+)}{\Pr(\mathcal{C}^-)} + \log \frac{\Pr(\mathbf{x} \mid \mathcal{C}^+)}{\Pr(\mathbf{x} \mid \mathcal{C}^-)}\right] \end{aligned}$$

In the cases of *bernoulli* (3.1.3) and *multinomial* (3.1.4) naive Bayes, there exist $v \in \mathbb{R}$ and $\mathbf{w} \in \mathbb{R}^p$ such that $y = \operatorname{sign}(v + \mathbf{w}^\top \mathbf{x})$. In these two special cases, the decision boundary is a hyperplane (which doesn't happen for gaussian naive Bayes). For both of them v has the same value, and by noting $\mathbf{w}_b$ and $\mathbf{w}_m$ the weights for the bernoulli and the multinomial case respectively, we have

$$v = \log \frac{\Pr(\mathcal{C}^+)}{\Pr(\mathcal{C}^-)}, \qquad \begin{cases} \mathbf{w}_b = \log(\boldsymbol{\theta}^+ \odot (\mathbf{1} - \boldsymbol{\theta}^-)) - \log(\boldsymbol{\theta}^- \odot (\mathbf{1} - \boldsymbol{\theta}^+)) \\ \mathbf{w}_m = \log \boldsymbol{\theta}^+ - \log \boldsymbol{\theta}^- \end{cases}$$

The figure 3.1 compares the decision boundary of logistic regression to the one of gaussian naive Bayes.

3.2 Sparse naive Bayes

A sparse version of naive Bayes was introduced in 2019 [?]. They add a sparsity constraint in the Bernoulli and the multinomial optimization problems portrayed above. It imposes the weight vector $\mathbf{w}$ to have a number

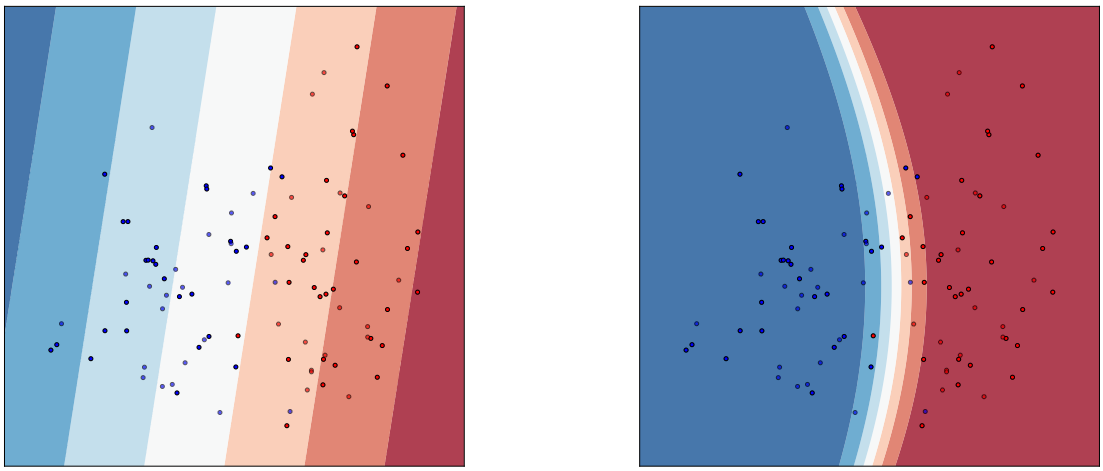

FIGURE 3.1: Illustration of the decision boundary for the logistic regression (left) and gaussian naive Bayes (right) for binary classification. For logistic regression, the separation is an hyperplane. For gaussian naive Bayes, the separation has a dependency on $x \odot x$ and can be non-linear. Both bernoulli and multinomial naive Bayes would have a linear separation.

of non-zero entries under a certain threshold $k \in \mathbb{N}$. That property is similar to the one of the Lasso presented in 1.1.3. But the sparsity in sparse naive Bayes (SNB) is controlled by an integer k which is the exact sparsity level desired on the weight vector, while the Lasso relies on a continuous penalty coefficient $\lambda \in \mathbb{R}$. This sparsity property makes SNB employable to perform feature selection, by keeping only the features whose weights are non-zero. Next sections detail the problem statement, the main results of the authors, and some applications.

3.2.1 Problem statement

Let $0 \leq k \leq p$ be a desired level of sparsity. We wish to train a naive Bayes classifier whose decision boundary depends on at most k features. For any $\mathbf{v} \in \mathbb{R}^d$ we note $\|\mathbf{v}\|_0$ the number of non-zero entries (the cardinality) of the vector. As shown in the previous section 3.1.5, there exist $v \in \mathbb{R}$ and $\mathbf{w} \in \mathbb{R}^p$ such that the prediction $y(\mathbf{x})$ for a new data point $\mathbf{x} \in \mathbb{R}^p$ is $\text{sign}(v + \mathbf{w}^\top \mathbf{x})$ for both bernoulli and multinomial naive Bayes. Furthermore, the jth entry of the decision vector $\mathbf{w}$ is null if and only if $\boldsymbol{\theta}_j^- = \boldsymbol{\theta}_j^+$, where $\boldsymbol{\theta}^-$ and $\boldsymbol{\theta}^+$ are the parameters of the loss functions $\mathcal{LL}_b$ and $\mathcal{LL}_m$ in 3.3 and 3.5 respectively. Naturally, imposing the constraint $\|\boldsymbol{\theta}^+ - \boldsymbol{\theta}^-\|_0 \leq k$ in the optimization problems will yield a weight vector $\mathbf{w}$ with the desired sparsity. The optimization problems for Bernoulli and multinomial SNB (shortened BSNB and MSNB respectively) can be phrased as follows:

$$\begin{aligned} \underset{\boldsymbol{\theta}^+, \boldsymbol{\theta}^-}{\text{maximize}} \quad & \mathcal{LL}_{\text{b}}\left(\boldsymbol{\theta}^+, \boldsymbol{\theta}^-\right) \\ & \quad = \mathbf{f}_+^\top \log \boldsymbol{\theta}^+ + (n_+ \mathbf{1} - \mathbf{f}_+)^\top \log\left(\mathbf{1} - \boldsymbol{\theta}^+\right) \\ & \quad\quad + \mathbf{f}_-^\top \log \boldsymbol{\theta}^- + (n_- \mathbf{1} - \mathbf{f}_-)^\top \log\left(\mathbf{1} - \boldsymbol{\theta}^-\right) \\ \text{subject to} \quad & \left\|\boldsymbol{\theta}^+ - \boldsymbol{\theta}^-\right\|_0 \leq k. \end{aligned} \tag{BSNB}$$

$$\begin{aligned} \underset{\boldsymbol{\theta}^+, \boldsymbol{\theta}^-}{\text{maximize}} \quad & \mathcal{LL}_{\text{m}}\left(\boldsymbol{\theta}^+, \boldsymbol{\theta}^-\right) = \mathbf{f}_+^\top \log \boldsymbol{\theta}^+ + \mathbf{f}_-^\top \log \boldsymbol{\theta}^- \\ \text{subject to} \quad & \left\|\boldsymbol{\theta}^+ - \boldsymbol{\theta}^-\right\|_0 \leq k \\ \text{and} \quad & \mathbf{1}^\top \boldsymbol{\theta}^+ = \mathbf{1}^\top \boldsymbol{\theta}^- = 1. \end{aligned} \tag{MSNB}$$

3.2.2 Main results and resolution

Surprisingly and despite the combinatorial constraints, these optimizations problems can be (approximately) solved very efficiently, with an additional minor cost compared to vanilla naive Bayes.

Especially for the Bernoulli case, an optimal solution can be computed in closed-form as shown in Theorem 5.

Theorem 5. *Suppose that $X \in \{0,1\}^{n \times p}$ is modeled by the Bernoulli distribution. Then, the exact solution to the problem BSNB can be computed. First, define $\mathbf{t}$ and $\mathbf{u}$ as follows*

$$\mathbf{t} = (\mathbf{f}^+ + \mathbf{f}^-) \odot \log\left(\frac{\mathbf{f}^+ + \mathbf{f}^-}{n}\right) + (n\mathbf{1} - \mathbf{f}^+ - \mathbf{f}^-) \odot \log\left(\mathbf{1} - \frac{\mathbf{f}^+ + \mathbf{f}^-}{n}\right)$$

$$\begin{aligned}\mathbf{u} = \mathbf{f}^+ \odot \log\frac{\mathbf{f}^+}{n^+} + (n^+\mathbf{1} - \mathbf{f}^+) \odot \log(\mathbf{1} - \frac{\mathbf{f}^+}{n^+}) \\ + \mathbf{f}^- \odot \log\frac{\mathbf{f}^-}{n^-} + (n^-\mathbf{1} - \mathbf{f}^-) \odot \log(\mathbf{1} - \frac{\mathbf{f}^-}{n^-})\end{aligned}$$

Let $\mathcal{I}$ be the set of $p-k$ smallest elements of $\mathbf{u} - \mathbf{t}$, and let

$$\theta^+_{\star\, j} = \theta^-_{\star\, j} = \frac{1}{n}(f^+_j + f^-_j)\ \forall j \in \mathcal{I}, \qquad \theta^\pm_{\star\, j} = \frac{f^\pm_j}{n^\pm}\ \forall j \notin \mathcal{I}$$

Forming the vectors $\mathbf{f}^-$ and $\mathbf{f}^+$ is very quick and takes asymptotically $\mathcal{O}(n \cdot p)$ summations. Then, constructing $\mathbf{t}$ and $\mathbf{u}$ can be done in $\mathcal{O}(p)$ calculations. Finding the k largest elements of $\mathbf{u} - \mathbf{t}$ takes $\mathcal{O}(p \cdot \log k)$ steps. Finally, constituting $\boldsymbol{\theta}^\star_\pm$ requires $\mathcal{O}(p)$ operations. In total, the maximizer can be found in $\mathcal{O}(n \cdot p + p \cdot \log k)$ steps, which is very close to the cost $\mathcal{O}(n \cdot p)$ of naive Bayes.

In the multinomial case, there is no closed-form solution, but a near-optimal one can be obtained as stated in Theorem 6.

Theorem 6. *Suppose that $X \in \mathbb{R}_+^{n \times p}$ is modeled by the multinomial distribution. Then the dual of MSNB can be solved very efficiently and a good solution to the primal can be recovered. Define $\phi_k : \alpha \mapsto s_k(\mathbf{h}(\alpha)) + C$ where C is some constant, s_k the sum of the k largest values of a vector, and*

$$\begin{aligned}\mathbf{h}(\alpha) = \mathbf{f}_+ \odot \log \mathbf{f}_+ + \mathbf{f}_- \odot \log \mathbf{f}_- - (\mathbf{f}_+ + \mathbf{f}_-) \odot \log(\mathbf{f}_+ + \mathbf{f}_-) \\ - \mathbf{f}_+ \log \alpha - \mathbf{f}_- \log(1 - \alpha)\end{aligned}$$

ϕ_k is the dual of MSNB and can be minimized very quickly (with bisection for example) as it is a one-dimensional convex function. Let $\alpha^\star$ be its minimizer, $\mathcal{I}$ the set of the $p-k$ smallest entries of $\mathbf{h}(\alpha^\star)$, and $B_\pm = \sum_{j \notin \mathcal{I}} f^\pm_i$. A primal point can be reconstructed as follows:

$$\theta^+_{\star\, j} = \theta^-_{\star\, j} = \frac{f^+_j + f^-_j}{\mathbf{1}^\top(\mathbf{f}^+ + \mathbf{f}^-)}\ \forall j \in \mathcal{I}, \qquad \theta^\pm_{\star\, j} = \frac{B_+ + B_-}{B_\pm}\frac{f^\pm_j}{\mathbf{1}^\top(\mathbf{f}^+ + \mathbf{f}^-)}\ \forall j \notin \mathcal{I}$$

Furthermore, it holds that $\psi(k-4) \leq \phi(k) \leq \psi(k) \leq \phi(k+4)$, implying that the duality gap is small if $\psi(k) - \psi(k-4)$ is small.

Experimentally, the duality gap quickly converges to 0 as k increases, and the reconstructed primal point is near-optimal. The time complexity is once again $\mathcal{O}(n \cdot p + p \cdot \log k)$, which is a minor additional cost compared to plain naive Bayes.

The authors experiment SNB on several text datasets, including AMZN, IMDB, TWTR, MPQA and SST2. They compare it with more costly methods like the Lasso, ℓ_1-penalized logistic regression and SVMs. They obtain competitive test accuracies, while training their models several order of magnitude faster.

3.3 Applications

The apparent low complexity of sparse naive Bayes compared to ℓ_1-penalized methods such as Lasso, logistic regression or SVMs makes is appealing for very large scale datasets. We mention here a few applications to which we will come back later[1].

3.3.1 Criteo dataset

As part of a Kaggle competition *Display Advertising Challenge*[2] in mid-2014, CriteoLabs shared log data collected over one week[3] whose features were undisclosed for confidentiality purposes. Main characteristics of this dataset can be found in Table 3.1. It consists in $\approx$ 45 millions of display ads with 39 features, and

[1] An implementation of sparse naive Bayes can be found here.
[2] The Kaggle competition can be found at this address.
[3] The competition's dataset can be downloaded at this address.

Samples	Total features	Numerical features	Categorical features	Features after encoding
45 840 617	39	13	26	33 762 590

TABLE 3.1: Criteo dataset characteristics. Even though the number of features is small, most categorical features have millions of categories. It makes the training of predicting models particularly challenging as it requires several dozens of GB of memory.

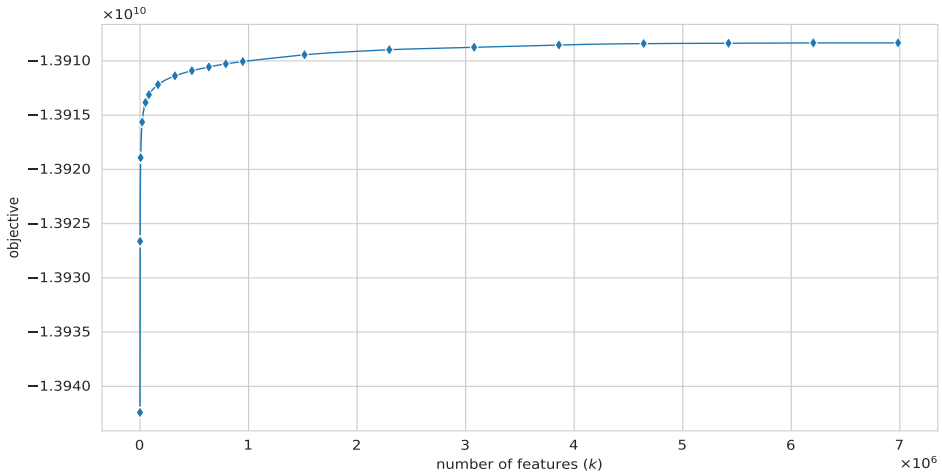

FIGURE 3.2: Optimal value of the MSNB optimization problem on the Criteo dataset as a function of the sparsity parameter k. Only 1–2 millions of the features explain most of the target vector (elbow heuristic).

a boolean label describing whether or not the ad was clicked by a customer. Among these 39 features, 26 are categorical and a classical one-hot encoding would end up in millions of features. This makes the Criteo dataset challenging, as it doesn't fit in the random-access memory after encoding, and potentially not in the mass storage of a standard computer either. Even on a small subset of the features, say 10%, selecting important features using Lasso or ℓ_1-penalized logistic regression isn't realistic. One-hot encoding isn't adapted to that situation. Another approach would be to one-hot encode for each categorical feature only the most frequent categories, and put the rest in a category *other*. The winners of the Kaggle competition used the hashing trick. It consists in choosing an encoding space size m, for example $m = 2^{20}$, and defining some hash function h: Categories $\to \{0, \dots, m-1\}$. This approach has several notable advantages. First, the final encoded feature space m can be adapted depending on the needs and on the computing power. It may be used in an online fashion without a first pass (that would be required for one-hot encoding in order to figure out all the existing categories). Lastly, it naturally handles new labels in the test set that were unseen in the train set (which would typically need a special *other* category in the one-hot encoding setting). Collisions between categories are likely to happen, and even collisions between categories from different features.

We present here in Figure 3.2. $m = 2^{24} = 16777216$ All the computations are done on a standard workstation (16GB, Intel Core i7 3.60GHz × 8). Sparse naive Bayes requires data averages to run, i.e. the sums of the negative and of the positive points. This part is time consuming but once these sums are computed they can be reused for any sparsity level k. Using a light hash function and PyPy, they were obtained in around 20 minutes. Only roughly half of the 2^{24} hash features were hit by the hash function. Then, SNB optimum are computed for 1200 log-spaced points using a Python and NumPy implementation of SNB in 1 hour. In this situation, what makes SNB particularly appealing is the fact that at no moment we need to load the full dataset in memory. We are only computing data averages whose shape are much smaller than the full matrix. Note also that most tasks could even be distributed to speed up the computations.

3.3.2 Genetic and fMRI data

Finally, sparse naive Bayes appear as an attractive alternative to the Lasso for the statistics computation in the knockoff procedure 2.2.

Chapter 4

Fast statistics computation

The computation of statistics is a key step in the knockoff selection process described in section 2. As it has been discussed, a large variety of statistics control the FDR, provided that the *flip-sign* property 3 is satisfied.

However, the power can drastically change depending on how good the statistics are. We also suppose that the knockoffs were generated respecting the conditions.... Experimentally, it appears that for most tasks the coefficients of a cross-validated Lasso

In this chapter, we note $X \in \mathbb{R}^{n \times p}$ the original design matrix, and we suppose that the distribution $\tilde{\mathcal{X}} \mid \mathcal{X}$ could be computed. Let $\tilde{X} \in \mathbb{R}^{n \times p}$ be sampled from that knockoff distribution.

4.1 Lasso is good but slow

Experimentally, cross-validating a regressor (or classifier) against the data $\left[X, \tilde{X}\right]$, keeping its coefficients in absolute value, and aggregating them as described in 2.2.2 yields excellent results. Most classical regression models scale pleasantly as p increases; $\mathcal{O}(n \cdot p)$ for Lasso and logistic regression, $\mathcal{O}(n \cdot p^2)$ for SVMs. But when the number of features is several thousands or more,

4.2 Multi-stage procedures

A natural way to address the high number of features is to split the selection into several steps. If p is high, you may want to reduce the number of features by cutting a large proportion with a cheap and fast procedure first, and use more sophisticated methods afterwards on the remaining set of features. Multi-stage procedure [**?**]

4.3 Fast statistics

Chapter 5

Efficient knockoffs construction

As shown in Chapter 1, in the case of gaussian knockoffs $\tilde{X}$ can be sampled from a normal distribution $\mathcal{N}(\boldsymbol{v}, \Upsilon)$ whose parameters $\boldsymbol{v}$, Υ are formulated in 5.1.

$$\tilde{\mathcal{X}} \mid \mathcal{X} \sim \mathcal{N}(\boldsymbol{v}, \Upsilon), \qquad \text{where} \quad \begin{cases} \boldsymbol{v} = \mathcal{X} - \mathcal{X}\Sigma^{-1}\operatorname{diag}(\mathbf{s}) \\ \Upsilon = \operatorname{diag}(\mathbf{s})\left(2I_{p\times p} - \Sigma^{-1}\operatorname{diag}(\mathbf{s})\right) \end{cases} \tag{5.1}$$

More generally, the same parameters are derived by only imposing $\left[X, \tilde{X}\right]_{\text{swap}(S)}$ and $\left[X, \tilde{X}\right]$ to have equal first two moments for any subset $S \subseteq \{1, \dots, p\}$, rather than the same distribution. These formulas are valid for any vector $\mathbf{s} \in \mathbb{R}^p$ such that Ω is indeed a covariance matrix (semidefinite positive).

$$\Omega = \begin{bmatrix} \Sigma & \Sigma - \operatorname{diag}\mathbf{s} \\ \Sigma - \operatorname{diag}\mathbf{s} & \Sigma \end{bmatrix} \succeq \mathbf{0}_{2p\times 2p}$$

As shown in proposition 4, this matrix is semidefinite positive if and only if $\mathbf{0}_{p\times p} \preceq \operatorname{diag}\mathbf{s} \preceq 2\Sigma$. The first inequality clearly holds if all the entries of $\mathbf{s}$ are positive. It is however trickier to identify all vectors $\mathbf{s}$ satisfying the second one.

In this chapter, we show why the choice of $\mathbf{s}$ is important and how to efficiently find a good one in the high dimensional setting. In the remaining, we note Σ the true covariance and $\hat{\Sigma}$ its estimation from the samples X (be it empirical or derived from more sophisticated methods).

5.1 Equi-correlated knockoffs

A cheap solution

For any psd matrix $A \in \mathbb{R}^p$, $A + \alpha I_{p\times p}$ has the same eigenvalues as A, but shifted by α. It gives a fast and simple way to find a feasible $\mathbf{s}$.

$$s_j = 2\lambda_{\min}(\hat{\Sigma}) \quad \text{for all } 0 \leq j \leq p \tag{5.2}$$

The problem of finding the smallest eigenvalue $\lambda_{\min}(\hat{\Sigma})$ can be solved efficiently, using for instance a singular value decomposition which runs in $\mathcal{O}(p^3)$ steps [?].

Why it is not desirable

For large values of p, the minimum eigenvalue of $\hat{\Sigma}$ is likely to be very small, unless $\hat{\Sigma}$ has a substantially special structure. Let us analyze briefly the covariance of $\left[\mathcal{X}; \tilde{\mathcal{X}}\right]$ to understand why it is unprofitable for the selection procedure. If $\tilde{\mathcal{X}}$ is built according to 5.1, it satisfies

$$\begin{cases} \operatorname{cov}(\tilde{\mathcal{X}}_j, \tilde{\mathcal{X}}_{j'}) = \operatorname{cov}(\mathcal{X}_j, \mathcal{X}_{j'}) \text{ for all } j, j' \\ \operatorname{cov}(\mathcal{X}_j, \tilde{\mathcal{X}}_{j'}) = \operatorname{cov}(\mathcal{X}_j, \mathcal{X}_{j'}) \text{ for all } j \neq j' \\ \operatorname{cov}(\mathcal{X}_j, \tilde{\mathcal{X}}_j) = \operatorname{cov}(\mathcal{X}_j, \mathcal{X}_j) - s_j \text{ for all } j \end{cases}$$

First, $\tilde{\mathcal{X}}$ has the same internal covariance has $\mathcal{X}$, and two distinct original and knockoff features have the same covariance as the one of the two corresponding original features. It makes the knockoff features sufficiently close to the original features to fool the estimator computing the statistics. However, an original feature j and its corresponding knockoff have a covariance that is smaller the larger s_j is. If s_j is close to 0, $\mathcal{X}_j$ cannot be distinguished from $\tilde{\mathcal{X}}_j$ and the selection would suffer from a low power.

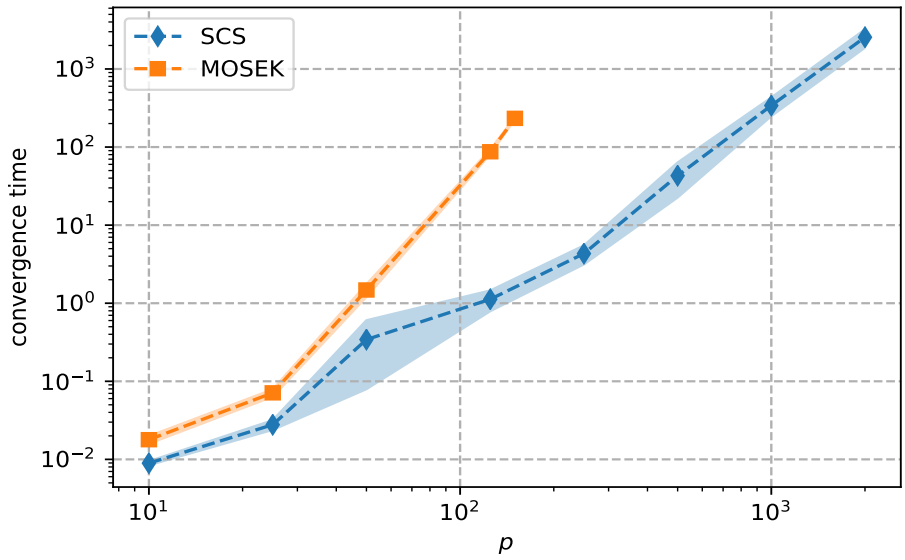

FIGURE 5.1: Time (in seconds) required to converge for SCS (blue) and MOSEK (orange) solvers as a function of p in log-log scale. Rapidly, MOSEK needs too much memory and can't be run for $p > 150$. Both match the theoretical $\mathcal{O}(p^3)$ rate and SCS already needs 15 minutes for $p = 1000$. See Appendix A.1 for details regarding the random generation of covariance matrices $\hat{\Sigma}$.

This fact is even more detrimental when p is large as $\lambda_{\min}$ is likely to be very close to 0. It motivates us to maximize the sum of the entries of $\mathbf{s}$ and it leads to solving a SDP.

5.2 SDP knockoffs

This observation motivates us to maximize the entries of $\mathbf{s}$, while maintaining the inequality $\operatorname{diag} \mathbf{s} \preceq 2\Sigma$. This can be formulated in the optimization problem 5.3.

$$\underset{\mathbf{s}\in\mathbb{R}^p}{\arg\max} \sum_{j=1}^{p} s_j \qquad \text{subject to } \begin{cases} s_j \geq 0 \text{ for all } j \\ \operatorname{diag} \mathbf{s} \preceq 2\hat{\Sigma} \end{cases} \tag{5.3}$$

This problem is a structured semidefinite program (SDP) and can efficiently be solved for small values of p by interior point methods [?] for example. For larger values of p, even first order methods like SCS [?] or alternating direction [?] quickly become intractable. In memory, it needs roughly $\mathcal{O}(m^2)$ where m is the number of constraints. Even though the convergence speed depends a lot on $\hat{\Sigma}$, it experimentally appears that alternative methods have to be considered when $p > 1000$. The Figure 5.1 shows the convergence time of the solvers SCS and MOSEK as a function of p. Theoretically, it takes $\mathcal{O}(p^3)$ and it is clear that when p is more than a few thousands. Moreover, most solutions returned by these algorithms are actually infeasible because of numerical approximations.

In order to reduce the computation time, Barber-Candès suggest to solve an approximated problem of 5.3 in 2 steps that we describe below.

Step 1. Pick an approximation $\hat{\Sigma}_{\text{approx}}$ of $\hat{\Sigma}$ and solve

$$\underset{\hat{\mathbf{s}}\in\mathbb{R}^p}{\arg\max} \sum_{j=1}^{p} \hat{s}_j \qquad \text{subject to } \begin{cases} \hat{s}_j \geq 0 \text{ for all } j \\ \operatorname{diag} \hat{\mathbf{s}} \preceq 2\hat{\Sigma}_{\text{approx}} \end{cases} \tag{5.4}$$

Step 2. Solve the one dimensional maximization

$$\underset{\gamma \in \mathbb{R}}{\arg\max}\ \gamma \quad \text{subject to} \quad \operatorname{diag}(\gamma \cdot \hat{\mathbf{s}}) \preceq 2\hat{\Sigma} \tag{5.5}$$

Finally, pick $\mathbf{s} = \gamma \cdot \hat{\mathbf{s}}$.

Remark 7. *Note that the two extreme options $\hat{\Sigma}_{approx} = \mathbf{I}$ and $\hat{\Sigma}_{approx} = \hat{\Sigma}$ yield the same solutions as solving equi-knockoffs 5.2 and full SDP knockoffs 5.3 respectively.*

Solving 5.5 in step 2 can be done very efficiently using bisection as it is a one-dimensional SDP. The optimization problem 5.4 is however the same as the one in 5.3, except for the approximation $\hat{\Sigma}_{\text{approx}}$. To speed up the computations, $\hat{\Sigma}_{\text{approx}}$ can be chosen to be a block-diagonal matrix of k blocks. The maximization then reduces to k smaller SDPs for which the solutions can be found more efficiently. These could even be distributed in several computation nodes. Picking $\hat{\Sigma}_{\text{approx}}$ is a compromise between available computation time and eventual power of the procedure. There is a priori no ideal way to to it. The block diagonal assumption depends on the order of the features. They should therefore be reordered ahead of time, typically by performing a clustering step to group very similar features.

5.3 A coordinate ascent approach

5.3.1 Notations and preliminaries

Notations. Let $M \in \mathbb{R}^{p \times p}$. Given two sets of indices $\mathcal{I}, \mathcal{J} \in \{1, \dots, p\}$, we note $M_{\mathcal{I},\mathcal{J}}$ the $|\mathcal{I}| \times |\mathcal{J}|$ matrix obtained by keeping the $|\mathcal{I}|$ rows and $|\mathcal{J}|$ columns indexed by $\mathcal{I}$ and $\mathcal{J}$ respectively. By convenience, an integer j denotes the set $\{j\}$ and j^c $\{1, \dots, p\} \setminus \{j\}$ in the matrix subscripting context. For example, if $M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$, then $M_{1^c,1^c} = \begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$ and $M_{1^c,1} = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$.

Positiveness characterisation. Suppose M is structured as

$$M = \begin{bmatrix} \xi & \mathbf{y}^\top \\ \mathbf{y} & B \end{bmatrix}$$

where B is symmetric and invertible.

Lemma 8. *Using Schur complements (see Appendix B.1), it holds that M is psd if and only if*

$$B \succeq \mathbf{0} \quad \textit{and} \quad \xi - \mathbf{y}^\top B^{-1} \mathbf{y} \geq 0$$

In subscript notations, this is equivalent to $M_{1^c,1^c} \succeq \mathbf{0}$ and $M_{1,1} - M_{1^c,1}^\top M_{1^c,1^c} M_{1^c,1} \geq 0$. This statement can be generalized as follows. For any $j \in \{1, \dots, p\}$, M is psd if and only if both conditions in 5.6 are satisfied

$$\begin{cases} M_{j^c,j^c} \succeq \mathbf{0} \\ M_{j,j} - M_{j^c,j}^\top M_{j^c,j^c} M_{j^c,j} \geq 0 \end{cases} \tag{5.6}$$

Proof. Let $j \in \{1, \dots, p\}$. Let P be the permutation matrix swapping columns $1 \leftrightarrow j$ and letting other columns unchanged. In two-line form, it is written

$$P = \begin{pmatrix} 1 & 2 & \dots & j-1 & j & j+1 & \dots & p \\ j & 2 & \dots & j-1 & 1 & j+1 & \dots & p \end{pmatrix}$$

M is psd if and only if $P^\top M P$ is psd. $P^\top M P$ is M with lines and columns 1 and j swapped. By applying Lemma 8 on it, we get the result. □

5.3.2 Coordinate ascent

The previous observation will be useful to characterize the feasibility of a potential solution $\mathbf{s}$ of the SDP 5.3. Using 5.6, it appears that the feasibility of $\mathbf{s}$ is equivalent to the three following conditions, for any $j \in$

$\{1, \dots, p\}$

$$2\hat{\Sigma} \succeq \operatorname{diag} \mathbf{s} \succeq \mathbf{0} \iff \begin{cases} \mathbf{s} \geq \mathbf{0} \\ 2\hat{\Sigma}_{j,j} - s_j - 4\hat{\Sigma}_{j^c,j}^\top \left(2\hat{\Sigma}_{j^c,j^c} - \operatorname{diag} \mathbf{s}_{j^c}\right)^{-1} \hat{\Sigma}_{j^c,j} \geq 0 \\ 2\hat{\Sigma}_{j^c,j^c} - \operatorname{diag} \mathbf{s}_{j^c} \succeq \mathbf{0} \end{cases} \tag{5.7}$$

This observation motivates the following coordinate approach, as described in [**?**]. Start with a feasible solution $\mathbf{s}_0$, for example $\mathbf{s}_0 = \mathbf{0}_p$. At each iteration, only one coordinate j of $\mathbf{s}$ will be updated by optimizing a sub-objective. As only the coordinates j changes, the constraints remain satisfied if we pick s_j to satisfy

$$\begin{cases} s_j \geq 0 \\ s_j \leq 2\hat{\Sigma}_{j,j} - 4\hat{\Sigma}_{j^c,j}^\top \left(2\hat{\Sigma}_{j^c,j^c} - \operatorname{diag} \mathbf{s}_{j^c}\right)^{-1} \hat{\Sigma}_{j^c,j} \end{cases} \tag{5.8}$$

that is, $s_j = \max\left(2\hat{\Sigma}_{j,j} - 4\hat{\Sigma}_{j^c,j}^\top \left(2\hat{\Sigma}_{j^c,j^c} - \operatorname{diag} \mathbf{s}_{j^c}\right)^{-1} \hat{\Sigma}_{j^c,j}, 0\right)$. Unfortunately, this method is not guaranteed to converge to the global maximum, even when the problem is concave. A slight modification depicted in the next section will save us.

5.3.3 Log-barrier

Instead of optimizing 5.3, we absorb the constraint $2\hat{\Sigma} \succeq \mathbf{s}$ into the objective by penalizing solutions $\mathbf{s}$ too close to the feasibility frontier, as described in §11.3 of [**?**]. It depends on an additional term $\lambda \cdot \log\det\left(2\hat{\Sigma} - \operatorname{diag} \mathbf{s}\right)$ for some coefficient $\lambda > 0$:

$$\underset{\mathbf{s} \in \mathbb{R}^p}{\arg\max} \sum_{j=1}^{p} s_j + \lambda \cdot \log\det\left(2\hat{\Sigma} - \operatorname{diag} \mathbf{s}\right) \qquad \text{subject to } s_j \geq 0 \text{ for all } j \tag{5.9}$$

where we consider that $\log 0 = -\infty$. Intuitively, a solution $\mathbf{s}$ such that the minimum eigenvalue of $2\hat{\Sigma} - \operatorname{diag} \mathbf{s}$ gets too close to 0 will be penalized by this log term.

In order to perform coordinate ascent, we are going to use the following lemma.

Lemma 9. *If* $M = \begin{bmatrix} \xi & \mathbf{y}^\top \\ \mathbf{y} & B \end{bmatrix}$, *then*

$$\begin{aligned} \det(M) &= \det\left(\xi - \mathbf{y}^\top B^{-1} \mathbf{y}\right) \cdot \det(B) \\ &= \left(\xi - \mathbf{y}^\top B^{-1} \mathbf{y}\right) \cdot \det(B) \end{aligned}$$

Proof. It is an immediate consequence of the formula for the determinant of a block matrix given in Appendix B.3. □

With the subscript notations, this identity can be noted as $\det(M) = \left(M_{1,1} - M_{1^c,1}^\top M_{1^c,1^c}^{-1} M_{1^c,1}\right) \cdot \det(M_{1^c,1^c})$. Employing the same idea as in 5.6, it can further be generalized to

$$\det(M) = \left(M_{j,j} - M_{j^c,j}^\top M_{j^c,j^c}^{-1} M_{j^c,j}\right) \cdot \det(M_{j^c,j^c})$$

for any $j \in \{1, \dots, p\}$.

By applying it to $\log\det\left(2\hat{\Sigma} - \operatorname{diag} \mathbf{s}\right)$, we get that for any $j \in \{1, \dots, p\}$,

$$\log\det\left(2\hat{\Sigma} - \operatorname{diag} \mathbf{s}\right) = \log\left(2\hat{\Sigma}_{j,j} - s_j - 4\hat{\Sigma}_{j^c,j}^\top Q_j^{-1} \hat{\Sigma}_{j^c,j}\right) + \log\det(Q_j)$$

where $Q_j = 2\hat{\Sigma}_{j^c,j^c} - \operatorname{diag} \mathbf{s}_{j^c}$ does not depend on s_j. The function

$$\alpha \mapsto \alpha + \lambda \log\left(2\hat{\Sigma}_{j,j} - \alpha - 4\hat{\Sigma}_{j^c,j}^\top Q_j^{-1} \hat{\Sigma}_{j^c,j}\right)$$

is concave and setting its derivative to 0 yields that its maximum is

$$\alpha^\star = 2\hat{\Sigma}_{j,j} - 4\hat{\Sigma}_{j^c,j}^\top Q_j^{-1} \hat{\Sigma}_{j^c,j} - \lambda$$

Algorithm 2 Coordinate ascent with log-barrier

Input: $\hat{\Sigma}$, barrier coefficient λ, decay μ, $\mathbf{s}^{(0)} = \mathbf{0}_p$
$\mathbf{s} = \mathbf{s}^{(0)}$
repeat
  for $j = 1, \dots, p$ **do**
    $Q_j = 2\hat{\Sigma}_{j^c, j^c} - \operatorname{diag} \mathbf{s}_{j^c}$
    $s_j = \max\left(2\hat{\Sigma}_{j,j} - 4\hat{\Sigma}_{j^c,j}^\top Q_j^{-1} \hat{\Sigma}_{j^c,j} - \lambda,\, 0\right)$
  end for
  $\lambda = \mu \cdot \lambda$
until stopping criteria

The jth coordinate is therefore updated as $s_j \leftarrow \max(\alpha^\star, 0)$. Note in particular that this update maintains the feasibility conditions 5.8 for any value of λ. Section 11.3 of [?] suggests that picking $\lambda = \epsilon/p$ will yield an ϵ-optimal solution. In practice, an initial λ_0 is picked and after each coordinate cycle it is multiplied by a decay rate $0 < \mu < 1$. The choice of μ is a trade-off.

This log-barrier algorithm is summarized in Algorithm 2 in pseudo-code. It can be proven to converge to the maximum of the modified objective 5.9. It is indeed a particular case of Theorem 3 in [?], which applies in this case as the constraints are simple, i.e. $\mathbf{s} \geq \mathbf{0}$ is of the form $L \leq \operatorname{diag} \mathbf{s} \leq U$.

The bottleneck of Algorithm 2 is the inversion of the matrix $Q_j \in \mathbb{R}^{(p-1)\times(p-1)}$ which takes $\mathcal{O}\left(p^3\right)$ steps. As it is done in every inner iteration, the algorithm's time complexity is $\mathcal{O}\left(n_{\text{iters}} \cdot p^4\right)$ when implemented naively.

5.4 Low-rank covariance approximation

Covariance estimation in the high-dimensional setting ($p > n$ where n is the number of samples) is challenging as the sample (empirical) covariance is not accurate. On top of that, if p is larger than $10\,000 - 20\,000$ it becomes challenging to solely store the covariance in the memory of a standard computer. Hopefully, big data matrices and especially correlation matrices often have an underlying low-rank structure, or at least can be approximated adequately by such a low-rank estimate [?]. An intuitive explanation is that there is only a small number of latent features explaining most of the data.

In this section, we suppose that the estimated covariance matrix $\hat{\Sigma}$ has the following structure

$$\hat{\Sigma} = D + U \Lambda U^\top \tag{5.10}$$

where $U \in \mathbb{R}^{p\times k}$ satisfies $U^\top U = I_k$, and both $D \in \mathbb{R}^{p\times p}$, $\Lambda \in \mathbb{R}^{k\times k}$ are diagonal and psd. k is typically much lower than p to save has much memory and computations as possible. Even in the (likely) case where the covariance matrix is not exactly diagonal plus low-rank, this approximating structure is general enough to yield satisfying results. In [?] the authors study this covariance estimation setting and provide an efficient algorithm to build it.

5.4.1 Low-rank coordinate ascent

The complexity of Algorithm 2 can be drastically reduced to $\mathcal{O}\left(n_{\text{iters}} \cdot p \cdot k^2\right)$ by taking advantage of the special structure of $\hat{\Sigma}$. Note $V = U\sqrt{\Lambda}$ so that $\hat{\Sigma} = D + VV^\top$. Then,

$$\begin{aligned} Q_j &= 2\hat{\Sigma}_{j^c, j^c} - \operatorname{diag} \mathbf{s}_{j^c} \\ &= 2D_{j^c} - \operatorname{diag} \mathbf{s}_{j^c} + 2V_{j^c,:}V_{j^c,:}^\top \end{aligned}$$

By noting $F = 2D_{j^c} - \operatorname{diag} \mathbf{s}_{j^c}$ and using the Woodbury formula,

$$Q_j^{-1} = F^{-1} - 2F^{-1}V_{j^c,:}\left(I_k + V_{j^c,:}^\top F^{-1} V_{j^c,:}\right)^{-1} V_{j^c,:}^\top F^{-1}$$

It gives that in the inner loop, s_j has to be updated with the quantity $\max(\alpha^\star, 0)$ where

$$\alpha^\star = \underbrace{2\hat{\Sigma}_{j,j} - 4\hat{\Sigma}_{j^c,j}^\top F^{-1} \hat{\Sigma}_{j^c,j} - \lambda}_{(*)} + \underbrace{8\hat{\Sigma}_{j^c,j}^\top F^{-1} V_{j^c,:}\left(I_k + V_{j^c,:}^\top F^{-1} V_{j^c,:}\right)^{-1} V_{j^c,:}^\top F^{-1} \hat{\Sigma}_{j^c,j}}_{(**)}$$

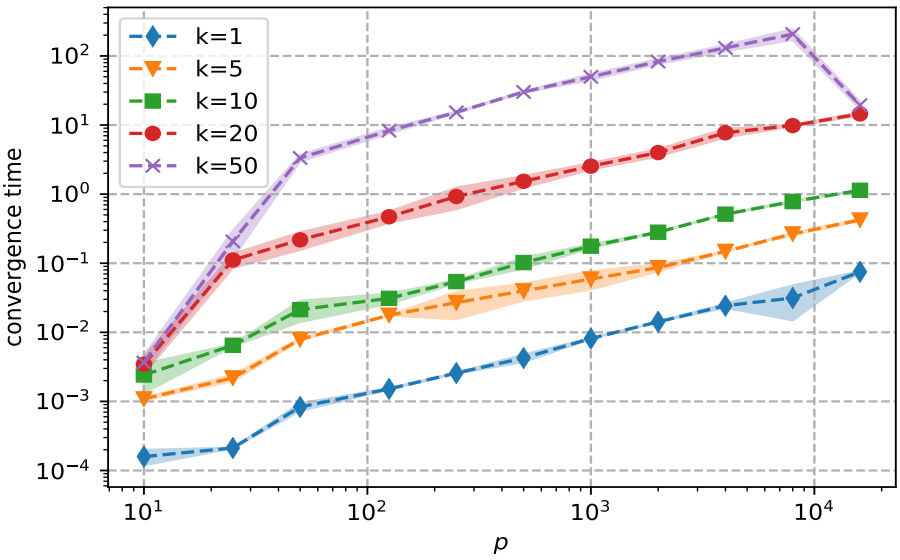

FIGURE 5.2: Convergence times of the low-rank coordinate ascent algorithm 3 as a function of p and k. It confirms the theoretical $\mathcal{O}(n_{\text{iters}} \cdot p \cdot k^3)$ rate. In practice, the algorithm converges much faster to an appropriate solution when using a larger tolerance threshold. See Appendix A.2 for details regarding the random data generation.

The term $(*)$ can easily be evaluated, but computing Q_j and its inverse in every inner loop would cost $\mathcal{O}\left(k^3\right)$ steps because of the inversion of the $k \times k$ matrix $M_j = \left(I_k + V_{j^c,:}^\top F^{-1} V_{j^c,:}\right)^{-1}$ in the term $(**)$. It can be reduced to $\mathcal{O}\left(k^2\right)$ using Sherman–Morrison rank-1 updates or, for a better numerical stability, $LDL^\top$ Cholesky rank-1 updates. To see this, note A the matrix

$$A = I_k + V^\top F^{-1} V$$

(which depends on $\mathbf{s}$ as F does). Then it can be observed that $M_j = A - F_{j,j} V_{j,:} V_{j,:}^\top$, so its inverse can simply be computed with a rank-1 update of the one of A.

Algorithm 3 Low-rank coordinate ascent

Input: approximation $\hat{\Sigma} = D + U\Lambda U^\top$, barrier coefficient λ, decay μ, $\mathbf{s}^{(0)} = \mathbf{0}_p$
$\mathbf{s} = \mathbf{s}^{(0)}$
$V = U\sqrt{\Lambda}$
$F = 2D - \operatorname{diag} \mathbf{s}$
$A = I_k + V^\top F^{-1} V$
repeat
  for $j = 1, \ldots, p$ **do**
    $Q_j = 2\hat{\Sigma}_{j^c,j^c} - \operatorname{diag} \mathbf{s}_{j^c}$
    $z = 2\hat{\Sigma}_{j,j} - 4\hat{\Sigma}_{j^c,j}^\top Q_j^{-1} \hat{\Sigma}_{j^c,j} - \lambda$
    $s_j = \max(z, 0)$
  end for
  $\lambda = \mu \cdot \lambda$
until stopping criteria

This scheme is summarized in Algorithm 3. It uses at most $\mathcal{O}(p \cdot k)$ memory (F never has to be fully computed as it's a diagonal) and the time complexity is $\mathcal{O}\left(n_{\text{iters}} \cdot p \cdot k^2\right)$, as stated above. However, numerical instabilities led us to perform some experiments in $\mathcal{O}\left(n_{\text{iters}} \cdot p \cdot k^3\right)$. Figure ?? shows the convergence time for Algorithm 3 as a function of p and k.

5.4.2 Efficient low-rank sampling

In this section, we detail briefly how to sample $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\upsilon}, \Upsilon)$ once a feasible $\mathbf{s}$ is computed, and in the special case where $\hat{\Sigma} = D + U\Lambda U^\top$. A classical approach to sample from a multivariate normal distribution is to sample first a vector $\tilde{\mathbf{z}} \in \mathbb{R}^p$ from $\mathcal{N}(0, 1)$, and then pose $\mathbf{z} = \boldsymbol{\upsilon} + L\tilde{\mathbf{z}}$, where L is a lower Cholesky factorization of Υ. This uses the fact that if $\mathbf{x} \sim \mathcal{N}\big(\mu, \Sigma\big)$, then

$$A\mathbf{x} + \mathbf{b} \sim \mathcal{N}\big(A\boldsymbol{\mu} + \mathbf{b},\, A\Sigma A^\top\big) \tag{5.11}$$

Note that this scheme works even if L is not lower-triangular, but only satisfies $LL^\top = \Upsilon$.

As Υ is a $p \times p$ matrix we may not want to store it in memory, nor compute a Cholesky factorization (which takes $\mathcal{O}\left(p^3\right)$ steps). We show here how to factorize Υ in a cheap way, requiring only $\mathcal{O}\left(k \cdot p\right)$ memory and $\mathcal{O}\left(k \cdot p^2\right)$ operations. As shown in Section 2.1.2, $\Upsilon = 2S - S\hat{\Sigma}^{-1}S$ where we note $S = \operatorname{diag} \mathbf{s}$. Using the low-rank structure $\hat{\Sigma} = D + U\Lambda U^\top$, the Sherman–Morrison–Woodbury formula (see Appendix B.2) gives

$$\begin{aligned}\hat{\Sigma}^{-1} &= (D + UU^\top)^{-1} \\ &= D^{-1} - D^{-1}U(I_k + U^\top D^{-1}U)^{-1}U^\top D^{-1}\end{aligned}$$

Note $L \in \mathbb{R}^{k\times k}$ the lower Cholesky factorization of $(I_k + U^\top D^{-1}U)^{-1}$ (which can be computed efficiently if k is small), $V = SD^{-1}UL \in \mathbb{R}^{p\times k}$, and $C = 2S - SD^{-1}S$. Then the covariance reduces to

$$\Upsilon = C + VV^\top$$

where C is diagonal and $VV^\top$ has rank at most k. C is not necessarily psd, which will force us to sample from a complex normal distribution. Let H be a complex square-root of C (thus, potentially with imaginary numbers on the diagonal), and $P = H^{-1}V \in \mathbb{R}^{p\times k}$. Then,

$$\Upsilon = H\left(I_{p\times p} + PP^\top\right)H^\top$$

$\left(I_{p\times p} + PP^\top\right)$ can be factorized in the following way. Note

$$W = \left(I_{k\times k} + \sqrt{I_{k\times k} + P^\top P}\right)^{-1} \in \mathbb{R}^{k\times k}$$

where the square-root is a $k \times k$ matrix root (which can again be computed efficiently if k is small). Then

$$I_{p\times p} + PP^\top = BB^\top, \qquad \text{where} \quad B = I_{p\times p} + PWP^\top$$

Finally, we note $M = HB$ and we have that $\Upsilon = MM^\top$. B is $p \times p$ but we never have to fully evaluate it; instead only W and P can be stored and matrix multiplications are then at most $p \times k$.

M is a complex matrix so we cannot directly use the property 5.11. But if $\mathcal{X} \sim \mathcal{N}(\mathbf{0}, I)$ and $\mathcal{Y} = i\mathcal{X}$, then $\operatorname{Im}(M\mathcal{Y}) \sim \mathcal{N}(\mathbf{0}, MM^\top = \Upsilon)$. Using this scheme, with $p = 15\,000$ and $k = 100$ we can draw $n = 5\,000$ samples in ≈ 4 seconds, while the naive method not taking into account the low rank approximation would need ≈ 180 seconds. In comparison sampling $5000 \times 15\,000$ numbers from $\mathcal{N}(0, 1)$ takes ≈ 2 seconds.

Chapter 6

Experiments

In this chapter, we show experiments

6.1 Setup

6.2 Results

Conclusion

In this master thesis, we investigated high-dimensional feature selection in the gaussian knockoffs framework setting. In high dimension, nearly all the steps of the selection procedure become challenging, and in particular the construction of the knockoff features, and the computation of statistics associated to each aggregated feature.

Building knockoffs inducing a high statistical power relies on the optimization of a semidefinite program. Hopefully, the structure and the simple bounds of this SDP

Appendix A

Data generation details

A.1 SCS and MOSEK convergence times

Experiments were run in Python using CVX [?] through the Python wrapper CVXPY [?]. Table A.1 shows the values of p that were used. We restrained the parameter p for MOSEK as it didn't scale as well as SCS. For each p, 5 replications were performed with matrices of the form $M = D + U\Lambda U^\top$ where D is diagonal,

SCS	MOSEK
10	10
25	25
50	50
125	125
250	150
500	
1000	
2000	

Table A.1: Orders of matrices used.

$\operatorname{diag} D \sim \mathcal{U}[0, 1]$, $U \in \mathbb{R}^{p\times k}$ for $k = 1, 5, 10, 20, 50$ are orthonormal random matrices, and $\Lambda \in \mathbb{R}^{k\times k}$ is diagonal and $\operatorname{diag} \Lambda^2 \sim \mathcal{U}[0, 1]$. No significant difference was measured when changing k, even for large values. The convergence is however much slower if D is scaled up.

A.2 Coordinate ascent

Appendix B

Linear algebra notes

In this section, we state and prove some useful linear algebra facts.

B.1 Schur complement and positive definiteness

Let $X = \begin{bmatrix} A & B^\top \\ B & C \end{bmatrix}$ be a symmetric matrix where A and C are invertible.

Definition 6. *The Schur complements $X_{/A}$ and $X_{/C}$ are defined as follows:*

$$X_{/A} = C - BA^{-1}B^\top, \qquad X_{/C} = A - B^\top C^{-1}B$$

Schur complements give a way to characterize the fact that X is positive definite and positive semidefinite.

Lemma 10. *The following three properties are equivalent:*

1. $X \succ \mathbf{0}$
2. $A \succ \mathbf{0}$ *and* $X_{/A} \succ \mathbf{0}$
3. $C \succ \mathbf{0}$ *and* $X_{/C} \succ \mathbf{0}$

As well as the three following:

1. $X \succeq \mathbf{0}$
2. $A \succ \mathbf{0}$ *and* $X_{/A} \succeq \mathbf{0}$
3. $C \succ \mathbf{0}$ *and* $X_{/C} \succeq \mathbf{0}$

Proof. X can be factorized as follows

$$X = \begin{bmatrix} I & \mathbf{0} \\ B^\top C^{-1} & I \end{bmatrix}^\top \begin{bmatrix} A - B^\top C^{-1}B & \mathbf{0} \\ \mathbf{0} & C \end{bmatrix} \begin{bmatrix} I & \mathbf{0} \\ B^\top C^{-1} & I \end{bmatrix}$$

which means that $X = Q^\top DQ$ where D is a block-diagonal matrix. Therefore $X \succ \mathbf{0}$ (resp. $\succeq \mathbf{0}$) if and only if $D \succ \mathbf{0}$ (resp. $\succeq \mathbf{0}$), which is equivalent to its diagonal blocks to be $\succ \mathbf{0}$ (resp. $\succeq \mathbf{0}$). To get the same result with the complement $X_{/A} = C - BA^{-1}B^\top$, we use the factorization

$$X = \begin{bmatrix} I & \mathbf{0} \\ BA^{-1} & I \end{bmatrix} \begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & C - BA^{-1}B^\top \end{bmatrix} \begin{bmatrix} I & \mathbf{0} \\ BA^{-1} & I \end{bmatrix}^\top$$

□

Consider the particular partition $X = \begin{bmatrix} \xi & \mathbf{y}^\top \\ \mathbf{y} & B \end{bmatrix}$ where B is invertible. Then a direct application of the theorem above gives that

$$\begin{cases} X \succ \mathbf{0} \iff B \succ \mathbf{0} \text{ and } \xi > \mathbf{y}^\top B^{-1}\mathbf{y} \\ X \succeq \mathbf{0} \iff B \succ \mathbf{0} \text{ and } \xi \geq \mathbf{y}^\top B^{-1}\mathbf{y} \end{cases}$$

More results regarding Schur complements can be found in [**?**]. A generalization involving pseudo-inverses of A and C when they are singular is possible.

B.2 Sherman–Morrison–Woodbury formula

The Sherman-Morrison formula gives a way to calculate the inverse of the sum of an invertible matrix $M \in \mathbb{R}^{p\times p}$ and a rank-1 update $uv^\top$.

Lemma 11. *(Sherman-Morrison) Let $u, v \in \mathbb{R}^p$. Then,*

$$\left(M + uv^\top\right)^{-1} = M^{-1} - \frac{M^{-1}uv^\top M^{-1}}{1 + v^\top M^{-1}u}$$

It can be generalized to a rank-k update UV as follows.

Lemma 12. *(Woodbury) Let $U, V \in \mathbb{R}^{p\times k}$. Then,*

$$\left(M + UV^\top\right)^{-1} = M^{-1} - M^{-1}U\left(I + V^\top M^{-1}U\right)^{-1}V^\top M^{-1}$$

Proofs and further details regarding these formulas can be found in [**?**].

B.3 Block matrix determinant

Lemma 13. *Let M be partitioned as $M = \begin{bmatrix} P & Q \\ R & S \end{bmatrix}$ where P and S are nonsingular. Then,*

$$\begin{cases} \det M = \det(S)\det(P - QS^{-1}R) \\ \det M = \det(P)\det(R - RP^{-1}Q) \end{cases}$$

Proof. Note the following block-matrix identity

$$\begin{bmatrix} P & Q \\ R & S \end{bmatrix}\begin{bmatrix} I & Q \\ -S^{-1}R & I \end{bmatrix} = \begin{bmatrix} P - QS^{-1}R & Q \\ 0 & S \end{bmatrix}$$

The fact that the determinant of a block-triangular matrix is the product of the determinants of the diagonal blocks gives the result. A similar factorization holds for the second identity. □

Appendix C

Cython acceleration

Code 4 Full coordinate ascent implementation with Cython and BLAS

```
import numpy as np
cimport cython
cimport numpy as np
from scipy.linalg.cython_blas cimport dsyr, dsymv, ddot

@cython.boundscheck(False)
@cython.wraparound(False)
@cython.nonecheck(False)
cdef _full_rank(
    int p, double[:, ::1] Sigma, int max_iterations, double lam, double mu, double tol
):
    cdef double[::1] s = np.zeros(p)
    cdef double[:, ::1] A = np.linalg.inv(Sigma) / 2
    cdef double[::1] temp = np.zeros(p)
    cdef double[::1] A_entry = np.zeros(p)
    cdef double q, r, c, z, delta, kappa
    cdef char* up = 'U'
    cdef int inc = 1
    cdef double alpha = 1, beta = 0

    for i in range(max_iterations):
        for j in range(p):
            dsymv(up, &p, &alpha, &A[0, 0], &p, &Sigma[j, 0], &inc, &beta, &temp[0], &inc)
            q = (ddot(&p, &temp[0], &inc, &Sigma[j, 0], &inc)
                - 2 * temp[j] * Sigma[j, j] + A[j, j] * Sigma[j, j] * Sigma[j, j])
            r = (temp[j] - Sigma[j, j] * A[j, j]) ** 2 / A[j, j]
            c = 4 * (q - r)
            z = 2 * Sigma[j, j] - lam - c
            if z < 0:
                z = 0

            delta = s[j] - z
            kappa = -delta / (1 + delta * A[j, j])
            for k in range(p):
                if k > j:
                    A_entry[k] = A[k, j]
                else:
                    A_entry[k] = A[j, k]
            dsyr(up, &p, &kappa, &A_entry[0], &inc, &A[0, 0], &p)
            s[j] = z
        lam = mu * lam
    return s
```

Because of the overhead of Python, we opted to implement SDP solvers in Cython, using SciPy's BLAS interface for Cython. Code 4 is an example of implementation of Algorithm 2.