TP4 - Réseaux Euclidiens

04 Décembre 2020

Les TPs comportent une partie à faire sur feuille, ainsi qu'une partie implémentation.

L'objectif de ce TP est de manipuler réseaux euclidiens, et notamment de voir l'algorithme LLL permettant de réduire une base. Cet algorithme a de nombreuses applications (conception de cryptosystèmes post-quantiques, résolution d'équation modulaire, résolution du problème du sac à dos, ...).

> Les exercices 2 et 3 correspondent à votre projet final. Ce projet est à rendre pour le 18/12 et comptera dans la note de l'UE.

Préliminaires et notations

De manière générale, un réseau d'un espace vectoriel $\mathbb{E}$ est un sous groupe discret de $\mathbb{E}$. Pour simplifier les choses, considérons $\mathbb{E} = \mathbb{R}^n$.

Le réseau est défini par sa base $(b_i)_{1 \leq i \leq r}$ dans $\mathbb{R}^n$, avec $r \leq n$. En pratique, on considère souvent $n = r$ (ce sera le cas dans ce TP), et les réseaux sont dit "de rang plein".
On note $\Lambda = \{\sum_{i=1}^{n} x_i b_i,\ x_i \in \mathbb{Z}\}$ un tel réseau.

On s'intéresse ici aux réseaux euclidiens. Les vecteurs sont donc munis de la norme euclidienne, notée $||\cdot||$, telle que $||x|| = \sqrt{\sum_{i=1}^{n} x_i^2}$.

Le produit scalaire de deux vecteurs $x, y \in \Lambda$ est noté $x \cdot y = \sum_{i=1}^{n} x_i y_i$.

Une base est dite minimale si elle est composée des vecteurs représentant les minima successifs de réseau. Autrement dit, elle ne contient que les vecteurs les plus courts.

Un réseau donné peut disposer de plusieurs bases équivalentes. Dans ce TP, nous allons étudier un algorithme permettant depuis une base $\mathcal{B}$ dont la norme des coefficients est grande, de trouver une base réduite, avec des normes petites.

Les notations des bases peut varier en représentant les vecteurs par des lignes ou de colonnes. Dans ce TP, **chaque vecteur représente une ligne dans la base** (c'est également le cas dans `sage`, mais pas dans votre cours).

Théorème 1 (Théorème de Minkowsky) *Soit Λ un réseau de dimension n dans dans $\mathbb{R}^n$ de déterminant $\det(\Lambda)$. Soit $S \subset \mathbb{R}^n$ convexe et symétrique par rapport à l'origine, notons $vol(S)$ son volume. Si $vol(S) > 2^n \det(\Lambda)$, alors S contient un élément non nul de Λ.*

Ce théorème peut s'interpréter de la manière suivante : si on prend un sphère de volume supérieur à $2^n \det(\Lambda)$, il y aura un vecteur non nul du réseau à l'intérieur.

Exercice 1 : Réseau en dimension 2

Afin de s'approprier ces notions, commençons par manipuler des réseaux de dimension 2.

Soit $\Lambda \subset \mathbb{R}^2$ tel que $\Lambda = \{ax + by,\ a, b \in \mathbb{Z}\}$. Ici, la base du réseau est formée par x et y. On supposant $||x|| \leq ||y||$, l'algorithme suivant permet de trouver une base minimale en dimension 2 :

Algorithme 1 : `Gauss`(x, y)

$v_1 = x$ et $v_2 = y$;
$found = False$;
while *not found* **do**
  $m = \lceil \frac{v_1 \cdot v_2}{v_1 \cdot v_1} \rfloor$;
  $v_2 = v_2 - mv_1$;
  if $||v_1|| \leq ||v_2||$ **then**
    found = true;
  else
    `swap`(v_1, v_2);

1. Soit $\Lambda \subset \mathbb{R}^2$ généré par $x = (-17, 13)$ et $y = (-56, 43)$. Déroulez à la main les calculs permettant d'aboutir à une base minimale (vous pouvez effectuer les calculs sur machine si vous préférez, l'idée est que vous voyez l'impact de chaque étape de l'algorithme).
   L'algorithme devrait terminer en 4 itérations.
2. Implémentez et commentez cet algorithme. Résumez en quelques phrases son fonctionnement.
3. Prenons un cas d'application la décomposition d'un nombre premier $p \equiv 1 \bmod 4$ en somme de deux carrés. Ce résultat peut être démontré
   (a) Rappelez pourquoi $p \equiv 1 \bmod 4 \Rightarrow \exists b \in \mathbb{Z},\ b^2 \equiv -1 \bmod p$. Une telle valeur b peut être rapidement trouvée en suivant la même approche que dans l'algorithme de Miller-Rabin.
   (b) Considérant le réseau $\Lambda = \{(x, y) \in \mathbb{Z}^2 \mid y \equiv bx \bmod p\}$, montrez que le vecteur (x_0, y_0) tel que $p = x_0^2 + y_0^2$ appartient à Λ.
   (c) En considérant $p = 2097169$, trouvez la décomposition de p en somme de deux carrés à l'aide de l'algorithme de Gauss.

Exercice 2 : LLL

L'algorithme LLL permet de généraliser l'approche vue en dimension 2, et d'obtenir une base réduite.
Notons $\mathcal{B} = (b_i)$ et $\mathcal{B}^* = (b_i^*)$ sa forme Gram-Schmidt orthogonalisée. On a donc $\mathcal{B} = M\mathcal{B}^*$, où $M = (\mu_{i,j} = \frac{b_i \cdot b_j^*}{||b_j^*||^2})$.
Une base (b_i) est dite α-réduite si, pour $1/4 < \alpha < 1$, si :

(i) Pour tout $1 < j < i \leq n$, on a $b_i \cdot b_j^* \ \leq \frac{||b_j^*||^2}{2}$, autrement dit : $|\mu_{i,j}| \leq 1/2$
(ii) Pour tout $2 \leq i \leq n$, $\alpha||b_{i-1}^*||^2 \leq ||b_i^* + \mu_{i,i-1} \cdot b_{i-1}^*||^2$.

Pour $\alpha = 1$, on ne sait pas prouver que LLL termine en temps polynomial.

1. Montrez que la condition (ii) est équivalente à
$$||b_i^*||^2 \geq (\alpha - \mu_{i,i-1}^2)||b_{i-1}^*||^2$$
2. Considérez la matrice $X = \begin{bmatrix} -2 & 7 & 7 & -5 \\ 3 & -2 & 6 & -1 \\ 2 & -8 & -9 & -7 \\ 8 & -9 & 6 & -4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$.
   La base (x_1, x_2, x_3, x_4) est elle α-réduite pour un α ?
   (a) Calculez M et X^* tels que $X = MX^*$. La méthode `gram_schmidt` de l'objet `Matrix` sur sage vous fera sûrement gagner du temps.
   (b) Justifiez votre conclusion à l'aide de la définition précédente.
3. Notons $\beta = \frac{4}{4\alpha - 1}$ le paramètre auxiliaire. Il sert essentiellement à alléger les notations. Notons en particulier que $\beta > 4/3$ et $\beta^{-1} = \alpha - 1/4$.

Montrez que si $\mathcal{B} = (x_i)$ est une base α-réduite d'un réseau Λ dans $\mathbb{R}^n$, et que $x \in \Lambda$ est un vecteur non nul de ce réseau, alors

$$||b_1|| \leq \beta^{(n-1)/2}||x||$$

(a) En utilisant la définition d'une base α-réduite, montrez que pour $1 \leq i \leq n$, on a

$$||b_i^*||^2 \geq \beta^{-(i-1)}||b_1||^2$$

(b) En déduire la résultat souhaité.

4. Implémentez l'algorithme LLL prenant en entrée une base $(b_1, ..., b_n)$ du réseau $\Lambda \subset \mathbb{R}^n$ et un paramètre de réduction $\alpha \in \mathbb{R}$ tel que $1/4 < \alpha < 1$.
La sortie est une base α-réduite du réseau.

(a) Justifiez le fonctionnement de la fonction `reduce`.

(b) Justifiez le fonctionnement de la fonction `exchange`.

(c) Justifiez le fait de devoir décrementer k si la condition ligne 36 n'est pas vérifiée.

Algorithme 2 : LLLAlgorithm

```
1  /* Rend x_k presque orthogonal à x_ℓ et met à jour le GSO et les coefficients */
2  func reduce(k, ℓ)
3    if |μ_{k,ℓ}| > 1/2 then
4      x_k = x_k − ⌈μ_{k,ℓ}⌋ x_ℓ
5      for j = 1, ..., ℓ − 1 do
6        μ_{k,j} = μ_{k,j} − ⌈μ_{k,ℓ}⌋
7      μ_{k,ℓ} = μ_{k,ℓ} − ⌈μ_{k,ℓ}⌋

8  /* Échange la ligne k et k − 1 et met à jour la GSO et les coefficients en conséquence */
9  func exchange(k)
10   swap(x_{k−1}, x_k)
11   v = μ_{k,k−1}
12   δ = γ*_k + v² γ*_{k−1}
13   μ_{k,k−1} = v γ*_{k−1} / δ
14   γ*_k = γ*_k γ*_{k−1} / δ
15   γ*_{k−1} = δ
16   for j = 1, ..., k − 2 do
17     swap(μ_{k−1,j}, μ_{k,j})
18   for i = k + 1, ..., n do
19     tmp = μ_{i,k}
20     μ_{i,k} = μ_{i,k_1} − v μ_{i,k}
21     μ_{i,k−1} = μ_{k,k−1} μ_{i,k} + tmp
22   μ_{k,l} = μ_{k,l} − ⌈μ_{k,l}⌋

23 func LLL(B, α)
24   (x_1, ..., x_n) = copy(b_1, ..., b_n)
25   /* Calcul de la GSO, en stockant γ*_i = ||x*_i||² */
26   for i = 1, ..., n do
27     x*_i = x_i
28     for j = 1, ..., i − 1 do
29       μ_{i,j} = (x_i · x*_j) / γ*_j
30       x*_i = x*_i − μ_{i,j} x*_j
31     γ*_i = ||x*_i||²
32   k = 2
33   /* Réduction de la base */
34   while k ≤ n do
35     reduce(k, k − 1)
36     if γ*_k ≥ (α − μ²_{k,k−1}) γ*_{k−1} then
37       for ℓ = k − 2, ..., 1 do
38         reduce(k, ℓ)
39       k = k + 1
40     else
41       exchange(k)
42       if k > 2 then
43         k = k − 1
```

Exercice 3 : Application au problème du sac à dos

Le problème du sac à dos (*Knapsack problem* en anglais) est un problème d'optimisation combinatoire bien connu. Ici, nous allons nous intéresser à une de ses variantes appelée problème de la somme des sous-ensembles (*sub-set sums problem*).
Le concept est simple : on dispose d'un sac à dos capable de supporter un certains poids s, et un ensemble d'objet ayant chacun un poids. L'objectif est de savoir s'il est possible d'atteindre la capacité maximal du sac à dos.

On peut représenter ce problème de la manière suivante :

— L'ensemble des objets est noté $A = (a_1, ..., a_n)$, où a_i décrit le poids de l'objet i.

— La capacité du sac à dos est notée s.

— On cherche à savoir s'il existe un sous ensemble $I \subseteq \{1, ..., n\}$ tel que $s = \sum_{i \in I} a_i$.

On peut reformuler ce dernier point de la manière suivante : on cherche $X = (x_1, ..., x_n) \in \{0, 1\}^n$ tel que $\sum_{i=1}^{n} x_i a_i = s$.

Exemple : Si on a les poids $(4, 3, 9, 1, 12, 17, 19, 23)$ et que $W = 35$, $X = (0, 1, 0, 1, 1, 0, 1, 0)$ permet d'atteindre exactement la capacité du sac. On notera que $X = (0, 0, 0, 0, 1, 0, 0, 1)$ fonctionne aussi.

Dans le cas général, ce problème est NP-complet. En revanche de la cas de séquences supercroissantes (chaque élément est plus grand que la somme des éléments plus petit que lui) est décidable en temps polynomial.

En supposant qu'une solution existe, l'objectif de cet exercice est de trouver un moyen d'appliquer LLL pour résoudre ce problème.

1. À partir des éléments $(a_1, ..., a_n, s)$, construisez un base d'un réseau $\Lambda \subset \mathbb{R}^{n+1}$ de telle sorte que le vecteur solution $(x_1, ..., x_n, 0)$ soit un vecteur court du réseau.
   Cette question représente l'essentiel du travail. Ce problème est largement documenté en ligne, vous pouvez vous renseigner si vous êtes bloqué.
2. Résoudre le problème sur l'exemple suivant :
   A = [32771, 65543, 131101, 262187, 524387, 1048759, 2097523, 4195057, 8390143, 16780259, 33560539, 67121039, 134242091, 268484171, 536968403]
   et s = 891221976.

Si vous n'avez pas réussi à implémenter votre LLL, vous pouvez utiliser le méthode LLL de l'objet Matrix sur sage.