Lattice Basis Reduction techniques based on the LLL algorithm

Bal K. Khadka

Michigan Technological University, Houghton, Michigan 49931, USA

August 26 - August 30, 2015

Contents

Introduction

Lattice Basis:
Given m linearly independent vectors $B = (b_1, b_2, ..., b_m)^T$ in Euclidean $n-$space $\mathbb{R}^n$ where $m \leq n$, the lattice $\mathcal{L}$ generated by them is defined as $\mathcal{L}(B) = \{\sum x_i b_i | x_i \in \mathbb{Z}\}$. That is $\mathcal{L}(B) = \{x^T B \,|\, x \in \mathbb{Z}^m\}$.

Let $B = (b_1, b_2, ..., b_m)^T$ be a basis of $\mathcal{L} \subset \mathbb{R}^n$ and U be an integral unimodular matrix (an $m \times m$ integer matrix having determinant ± 1), then UB is another basis of $\mathcal{L}$.

Introduction

Lattice Basis:

Given m linearly independent vectors $B = (b_1, b_2, ..., b_m)^T$ in Euclidean $n-$space $\mathbb{R}^n$ where $m \leq n$, the lattice $\mathcal{L}$ generated by them is defined as $\mathcal{L}(B) = \{\sum x_i b_i | x_i \in \mathbb{Z}\}$. That is $\mathcal{L}(B) = \{x^T B \,|\, x \in \mathbb{Z}^m\}$.

Let $B = (b_1, b_2, ..., b_m)^T$ be a basis of $\mathcal{L} \subset \mathbb{R}^n$ and U be an integral unimodular matrix (an $m \times m$ integer matrix having determinant ± 1), then UB is another basis of $\mathcal{L}$.

Introduction

Minkowski Convex Body Theorem:

A convex set $S \subset \mathbb{R}^n$ which is symmetric about origin and with volume greater than $m2^n det(\mathcal{L})$ contains at least m non-zero distinct lattice pairs $\pm x_1, \pm x_2, \ldots . \pm x_m$.

Corollary:

If $\mathcal{L} \subset \mathbb{R}^n$ is an n dimensional lattice wih determinant $det(\mathcal{L})$ then there is a nonzero $b \in \mathcal{L}$ such that $\mid b \mid \leq \frac{2}{\sqrt{\pi}}[\Gamma(\frac{n}{2}+1)]^{\frac{1}{n}}(det(\mathcal{L}))^{\frac{1}{n}}$.

Introduction

Minkowski Convex Body Theorem:
A convex set $S \subset \mathbb{R}^n$ which is symmetric about origin and with volume greater than $m2^n det(\mathcal{L})$ contains at least m non-zero distinct lattice pairs $\pm x_1, \pm x_2, \ldots . \pm x_m$.

Corollary:
If $\mathcal{L} \subset \mathbb{R}^n$ is an n dimensional lattice wih determinant $det(\mathcal{L})$ then there is a nonzero $b \in \mathcal{L}$ such that $\mid b \mid \leq \frac{2}{\sqrt{\pi}}[\Gamma(\frac{n}{2}+1)]^{\frac{1}{n}}(det(\mathcal{L}))^{\frac{1}{n}}$.

Goal

Given an integral lattice basis of a lattice $\mathcal{L} \subset \mathbb{R}^n$ as input, to find a vector in the lattice $\mathcal{L}$ with a minimal Euclidean norm.

Gram-Schimdt Orthogonalization (GSO)

- Given a basis $B = (b_1, b_2, ..., b_m)^T$ for a vector space $\mathbb{R}^n$, we can use GSO process to construct an orthogonal basis $B^* = (b_1^*, b_2^*, ..., b_m^*)^T$ such that $b_1^* = b_1$ and
$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^* \quad (2 \leq i \leq m),$$
$$\mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \ (1 \leq j < i \leq n).$$

- $B = MB^*$, where $M = (\mu_{ij})$ is a lower triangular matrix.

- For any non zero lattice vector $b \in \mathcal{L} \subset \mathbb{R}^n$ we have $\| b \| \geq min\{\| b_1^* \|, \cdots, \| b_n^* \|\}$.

Gram-Schimdt Orthogonalization (GSO)

- Given a basis $B = (b_1, b_2, ..., b_m)^T$ for a vector space $\mathbb{R}^n$, we can use GSO process to construct an orthogonal basis $B^* = (b_1^*, b_2^*, ..., b_m^*)^T$ such that $b_1^* = b_1$ and
$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^* \quad (2 \leq i \leq m),$$
$$\mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \quad (1 \leq j < i \leq n).$$

- $B = MB^*$, where $M = (\mu_{ij})$ is a lower triangular matrix.

- For any non zero lattice vector $b \in \mathcal{L} \subset \mathbb{R}^n$ we have $\| b \| \geq min\{\| b_1^* \|, \cdots, \| b_n^* \|\}$.

Gram-Schimdt Orthogonalization (GSO)

- Given a basis $B = (b_1, b_2, ..., b_m)^T$ for a vector space $\mathbb{R}^n$, we can use GSO process to construct an orthogonal basis $B^* = (b_1^*, b_2^*, ..., b_m^*)^T$ such that $b_1^* = b_1$ and
$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^* \quad (2 \leq i \leq m),$$
$$\mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \quad (1 \leq j < i \leq n).$$
- $B = MB^*$, where $M = (\mu_{ij})$ is a lower triangular matrix.
- For any non zero lattice vector $b \in \mathcal{L} \subset \mathbb{R}^n$ we have $| \, b \, | \geq min\{| \, b_1^* \, |, \cdots, | \, b_n^* \, |\}$.

LLL Reduced Bases

- Let $\mathcal{L}(B)$ with $B = (b_1, b_2, ..., b_m)$ be a lattice in $\mathbb{R}^n$ with GSO vector $b_1^*, b_2^*, ..., b_m^*$. The basis B is called α reduced (or LLL-reduced with the reduction parameter $\alpha \in (\frac{1}{4}, 1)$) if the following conditions hold:

- a) $|\mu_{i,j}| \leq \frac{1}{2}$ for $1 \leq j < i \leq m$,

- b) $|b_i^* + \mu_{i,i-1} b_{i-1}^*|^2 \geq \alpha |b_{i-1}^*|^2$ for $2 \leq i \leq m$.

LLL Reduced Bases

- Let $\mathcal{L}(B)$ with $B = (b_1, b_2, ..., b_m)$ be a lattice in $\mathbb{R}^n$ with GSO vector $b_1^*, b_2^*, ..., b_m^*$. The basis B is called α reduced (or LLL-reduced with the reduction parameter $\alpha \in (\frac{1}{4}, 1)$) if the following conditions hold:

- a) $|\mu_{i,j}| \leq \frac{1}{2}$ for $1 \leq j < i \leq m$,

- b) $|b_i^* + \mu_{i,i-1} b_{i-1}^*|^2 \geq \alpha |b_{i-1}^*|^2$ for $2 \leq i \leq m$.

LLL Reduced Bases

- Let $\mathcal{L}(B)$ with $B = (b_1, b_2, ..., b_m)$ be a lattice in $\mathbb{R}^n$ with GSO vector $b_1^*, b_2^*, ..., b_m^*$. The basis B is called α reduced (or LLL-reduced with the reduction parameter $\alpha \in (\frac{1}{4}, 1)$) if the following conditions hold:
- a) $|\mu_{i,j}| \leq \frac{1}{2}$ for $1 \leq j < i \leq m$,
- b) $|b_i^* + \mu_{i,i-1} b_{i-1}^*|^2 \geq \alpha |b_{i-1}^*|^2$ for $2 \leq i \leq m$.

Lattice Diffusion and Sublattice Fusion Algorithm

- **Input:** Basis $B = (b)_{m \times n}$ of $\mathcal{L} \subset \mathbb{R}^n$, $\beta < m \to$ *Block Size* and N, $M \to$ parameters
- Take N permutation matrices P_j, $(1 \leq j \leq N)$ with radius close to m.
- $M \leftarrow \bigcup_{i=1}^{M} \beta_i \uparrow Sort\{\text{LLL}(P_j B) | \text{length of } (\text{LLL}(P_j B)) \text{ is minimum for } 1 \leq j \leq N\}$.
- $B' \leftarrow LLL(M)$
- **Output:** a vector of minimal length

Lattice Diffusion and Sublattice Fusion Algorithm

- **Input:** Basis $B = (b)_{m\times n}$ of $\mathcal{L} \subset \mathbb{R}^n$, $\beta < m \rightarrow$ *Block Size* and N, $M \rightarrow$ parameters
- Take N permutation matrices P_j, $(1 \leq j \leq N)$ with radius close to m.
- $M \leftarrow \bigcup_{i=1}^{M} \beta_i \uparrow Sort\{\text{LLL}(P_jB) | \text{length of } (\text{LLL}(P_jB)) \text{ is minimum for } 1 \leq j \leq N\}$.
- $B' \leftarrow LLL(M)$
- **Output:** a vector of minimal length

Lattice Diffusion and Sublattice Fusion Algorithm

- **Input:** Basis $B = (b)_{m \times n}$ of $\mathcal{L} \subset \mathbb{R}^n$, $\beta < m \rightarrow$ *Block Size* and N, $M \rightarrow$ parameters
- Take N permutation matrices P_j, $(1 \leq j \leq N)$ with radius close to m.
- $M \leftarrow \bigcup_{i=1}^{M} \beta_i \uparrow Sort\{\text{LLL}(P_j B) | \text{length of } (\text{LLL}(P_j B)) \text{ is minimum for } 1 \leq j \leq N\}$.
- $B' \leftarrow LLL(M)$
- **Output:** a vector of minimal length

Lattice Diffusion and Sublattice Fusion Algorithm

- **Input:** Basis $B = (b)_{m \times n}$ of $\mathcal{L} \subset \mathbb{R}^n$, $\beta < m \rightarrow$ *Block Size* and N, $M \rightarrow$ parameters
- Take N permutation matrices P_j, $(1 \leq j \leq N)$ with radius close to m.
- $M \leftarrow \bigcup\limits_{i=1}^{M} \beta_i \uparrow Sort\{\text{LLL}(P_j B) | \text{length of } (\text{LLL}(P_j B)) \text{ is minimum for } 1 \leq j \leq N\}$.
- $B' \leftarrow LLL(M)$
- **Output:** a vector of minimal length

Lattice Diffusion and Sublattice Fusion Algorithm

- **Input:** Basis $B = (b)_{m \times n}$ of $\mathcal{L} \subset \mathbb{R}^n$, $\beta < m \rightarrow$ *Block Size* and N, $M \rightarrow$ parameters
- Take N permutation matrices P_j, $(1 \leq j \leq N)$ with radius close to m.
- $M \leftarrow \bigcup_{i=1}^{M} \beta_i \uparrow Sort\{\text{LLL}(P_j B) | \text{length of } (\text{LLL}(P_j B)) \text{ is}$ minimum for $1 \leq j \leq N\}$.
- $B' \leftarrow LLL(M)$
- **Output:** a vector of minimal length

Hill Climbing Algorithm

- **Begin**: Basis $B = (b)_{m \times n}$ of $\mathcal{L} \subset \mathbb{R}^n$.
- Take k permutation matrices P_j, $(2 \leq j \leq k)$ such that $d(P_j, I_m) = r$ $(r \leq m)$, where d is a hamming distance.
- $B \leftarrow \{\text{LLL}(P_j B) | \text{length of } (\text{LLL}(P_j B)) \text{ is minimum for } 1 \leq j \leq k\}$.
- End if the desired bound is achieved, or no further improvement is observed.
- else,
- go to the step 2.

Hill Climbing Algorithm

- **Begin**: Basis $B = (b)_{m \times n}$ of $\mathcal{L} \subset \mathbb{R}^n$.
- Take k permutation matrices P_j, $(2 \leq j \leq k)$ such that $d(P_j, I_m) = r$ $(r \leq m)$. where d is a hamming distance.
- $B \leftarrow \{\text{LLL}(P_j B) | \text{length of } (\text{LLL}(P_j B)) \text{ is minimum for } 1 \leq j \leq k\}$.
- End if the desired bound is achieved, or no further improvement is observed.
- else,
- go to the step 2.

Hill Climbing Algorithm

- **Begin**: Basis $B = (b)_{m \times n}$ of $\mathcal{L} \subset \mathbb{R}^n$.
- Take k permutation matrices P_j, $(2 \leq j \leq k)$ such that $d(P_j, I_m) = r$ $(r \leq m)$. where d is a hamming distance.
- $B \leftarrow \{\text{LLL}(P_j B) | \text{length of } (\text{LLL}(P_j B)) \text{ is minimum for } 1 \leq j \leq k\}$.
- End if the desired bound is achieved, or no further improvement is observed.
- else,
- go to the step 2.

Hill Climbing Algorithm

- **Begin**: Basis $B = (b)_{m \times n}$ of $\mathcal{L} \subset \mathbb{R}^n$.
- Take k permutation matrices P_j, $(2 \leq j \leq k)$ such that $d(P_j, I_m) = r \ (r \leq m)$. where d is a hamming distance.
- $B \leftarrow \{\mathrm{LLL}(P_j B) | \text{length of } (\mathrm{LLL}(P_j B)) \text{ is minimum for } 1 \leq j \leq k\}$.
- End if the desired bound is achieved, or no further improvement is observed.
- else,
- go to the step 2.

Hill Climbing Algorithm

- **Begin**: Basis $B = (b)_{m \times n}$ of $\mathcal{L} \subset \mathbb{R}^n$.
- Take k permutation matrices P_j, $(2 \leq j \leq k)$ such that $d(P_j, I_m) = r$ $(r \leq m)$. where d is a hamming distance.
- $B \leftarrow \{\mathrm{LLL}(P_jB) | \text{length of } (\mathrm{LLL}(P_jB))$ is minimum for $1 \leq j \leq k\}$.
- End if the desired bound is achieved, or no further improvement is observed.
- else,
- go to the step 2.

Hill Climbing Algorithm

- **Begin**: Basis $B = (b)_{m \times n}$ of $\mathcal{L} \subset \mathbb{R}^n$.
- Take k permutation matrices P_j, $(2 \leq j \leq k)$ such that $d(P_j, I_m) = r$ $(r \leq m)$. where d is a hamming distance.
- $B \leftarrow \{\text{LLL}(P_j B) | \text{length of } (\text{LLL}(P_j B)) \text{ is minimum for } 1 \leq j \leq k\}$.
- End if the desired bound is achieved, or no further improvement is observed.
- else,
- go to the step 2.

Our experiment

- Case 1: we constructed hadamard matrices and inflated using integral unimodular matrices.
- Case 2: we picked Ideal lattices from online resources.
- We used Hill climbing/lattice diffusion and sublattice fusion algorithm to get the desired approximated shortest vector.
- We successfully reduced B to find the competitive shortest vectors.

Our experiment

- Case 1: we constructed hadamard matrices and inflated using integral unimodular matrices.
- Case 2: we picked Ideal lattices from online resources.
- We used Hill climbing/lattice diffusion and sublattice fusion algorithm to get the desired approximated shortest vector.
- We successfully reduced B to find the competitive shortest vectors.

Our experiment

- Case 1: we constructed hadamard matrices and inflated using integral unimodular matrices.
- Case 2: we picked Ideal lattices from online resources.
- We used Hill climbing/lattice diffusion and sublattice fusion algorithm to get the desired approximated shortest vector.
- We successfully reduced B to find the competitive shortest vectors.

Our experiment

- Case 1: we constructed hadamard matrices and inflated using integral unimodular matrices.
- Case 2: we picked Ideal lattices from online resources.
- We used Hill climbing/lattice diffusion and sublattice fusion algorithm to get the desired approximated shortest vector.
- We successfully reduced B to find the competitive shortest vectors.

Results

- Inflated Hadamard Matrix
- Ideal Lattice
- ASVP Hall of Fame

Results

- Inflated Hadamard Matrix
- Ideal Lattice
- ASVP Hall of Fame

Results

- Inflated Hadamard Matrix
- Ideal Lattice
- ASVP Hall of Fame

References

- Lattice Basis Reduction by Murray R. Bermner
- LLL reduction using NTL library by Victor Shoup
- http://www.latticechallenge.org/ideallattice-challenge/in-dex.php

The End

Any Question?

Thank You!

The End

Any Question?

Thank You!