# Lecture 18

# Lattices and the Shortest Vector Problem

December 3, 2004
Lecturer: Kamal Jain
Notes: Chris Ré

## 18.1 Overview

In this lecture, we will show first that every lattice has a basis that can be found in a polynomially amount of time from the original vector description. This allows us to assume that we are given a basis of a lattice as input. Then we will tackle a problem in lattices, the shortest vector problem.

## 18.2 Finding A Basis

### 18.2.1 Prelimiaries

First we state some easy lemmas and give some intuition for the theoreom. A lattice of vectors $v_1 \dots v_m$ will be denoted $\{v_i\}^m$ for convenience.

**Lemma 18.1.** *Two lattices $\{v_i\}^n$ and $\{u_j\}^m$ are the same $\Leftrightarrow$ each $v_i$ can be written as an integer combination of the $\{u_j\}^m$ and similarily each $u_j$ can be written as a combination of the $\{v_i\}^n$ .*

**Lemma 18.2.** *Greatest common divisor of a and b (gcd(a,b)) (w.log) $a < b$, is the same as gcd(a,b-a).*

*Proof.* Let $d = gcd(a, b)$, notice $d|a$ and $d|b - a$ so it is a divisor. If there were a divisor greater of $a, b - a$ then it would be a divisor of $b - a + a = b$, a contradiction. □

This lemma immediately gives euclid's algorithm for computing the gcd. This algorithm will be the intuition for much of this lecture.

**Lemma 18.3.** $gcd(a_1, \dots, a_i, \dots, a_j \dots, a_m) = gcd(a_1, \dots, a_i, \dots, a_j - a_i, \dots, a_m)$

*Proof.* same argument □

**Lemma 18.4.** *if $gcd(\{a_i\}) = d \rightarrow \exists x_i \ \sum x_i a_i = d$.*

*sketch.* Just think of euclid's algorithm, when it terminates you get $d = ax + y$. Walk backward through the execution of the algorithm, you will be able to write the coefficients of the original terms by subsitution. □

### 18.2.2 Lattice Basis Theorem

**Theorem 18.5.** *Let L be a lattice $\{u_i\}^m$. In polynomial time L can be written as $\{v_j\}^n$ where the $v_j$ are linearly independent.*

*Remark.* Notice this is a generalization of finding gcds of lists of numbers.

*Proof.* We may assume $\{u_i\}^m$ are linearly dependent, because otherwise we are done. By definition this means $\exists c_i \sum c_i v_i = 0$ such that not all $c_i = 0$.

Notice these $c_i$ are in $\mathbb{Q}$ because all numbers involved are rational. The $c_i$s are also of small (polynomial) size by a cramer's rule argument. We can set them up to form a matrix equation $Vc = 0$ and solve for c.

We now note some simplifying conditions. We can take $gcd(\{c_i\}) = 1$ because we can divide out any gcd from all the terms.

Each $c_i$ can be taken $\geq 0$. This is because we are dealing with an integer span and can swap the sign of vectors. This changes any negative $c_i$ to positive.

Notice that if any $c_i = 1$, we are done. This because $\sum_{j \neq i} c_j(-v_j) = v_i$. We can just drop $v_i$ from the list, since it is redundant. However there may be no such vector so that there is some $c_i = 1$. Consider for example $(3, 5)$.

We use our fact about gcds and try to reduce some $c_i$ to 1. Let $0 < c_i \leq c_j$. $c_i, c_j$ exist because there is a non-trivial combination, this gives us that one such number exists. Also it is the case that their sum is equal to 0, which means there must be at least one other non-zero coefficent.

We now perform the following substitution.

$$c_i v_i + \cdots + c_j v_j \cdots \rightarrow c_i(v_i + v_j) + \cdots + (c_j - c_i)v_j \ldots$$

Notice that we are decreasing the sum of the coefficient sizes and since they are integers we will terminate by well ordering. Like euclid we need to be careful to terminate in polynomial time. Notice, in an analgous way to euclid's algorithm we can take modulus of the smallest vector. This is sufficient but the proof is omitted.

Also notice that we have not changed the lattice by this substitution. Only one vector changed: $v_i = (v_i + v_j) + (-1)v_j$. This means that the span is still as large because we can recover $v_i$ and we have not enlarged it because $(v_i + v_j)$ was in the span of the original. □

*Remark.* It is crucial that we take the basis vectors to be rational. For example $\{1, \sqrt{2}\}$ is dense.

We will know assume that all lattices are described by their basis.

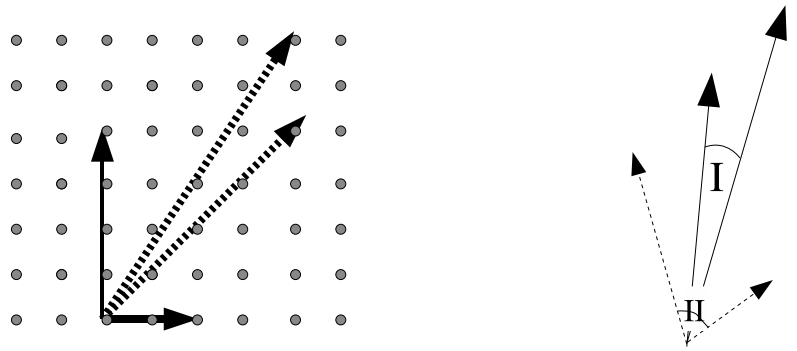
## 18.3 Shortest Vector Problem

Given a lattice with basis $\{v_i\}^n$, find the shortest non-zero vector with respect to the $l_2$ norm. Figure 18.1(a) shows why this is non-trivial, we could be given a very 'long' bases.

**Lemma 18.6.** *Suppose L has two different bases $\{u_i\}^n$ and $\{v_j\}^n$ then $|det(\{u_i\}^n)| = |det(\{u_i\}^n)|$*

*Remark.* We can take L to be in $R^n$ without loss, since that is the dimension of the lattice.

**Definition 18.1.** $det(\{u_i\}^n) = det \begin{bmatrix} u_{11} & u_{21} & \ldots \\ u_{12} & u_{22} & \ldots \\ \ldots & \ldots & \ldots \end{bmatrix}$




(a) A Lattice with two different bases (b) Reduction of angles between lattice vectors

Figure 18.1: Angles and Basis

*Proof.* Written as row vectors, we know that by our earlier lemma that we can write each basis in terms of the other concretely. $\begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{bmatrix} = M_1 \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix}$ and $\begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix} = M_2 \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{bmatrix}$

Substuiting we can write: $\begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{bmatrix} = M_1 M_2 \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{bmatrix}$ Therefore, $det(M_1)det(M_2) = 1$ and these matricies are over the integers, so this means their determinant is as well and $det(M_1) = det(M_2) = \pm 1$. This proves the claim. □

### 18.3.1 The method of Gauss for dimension 2

Now we are going to start with a method of Gauss to compute the shortest vector in $\mathbb{R}^2$. This will give the intuition for the method which works in higher dimensions. The algorithm looks very similiar to euclid's algorithm but is not a generalization.

Some of the intuition comes from the fact that $|det(v_1 \dots v_n)|$ is the volume of the parallelpiped you get when you draw them. Our goal is to shrink the angle between the two vectors $\theta$ in order to get them perpindicular. As we reduce we get $|v_1||v_2|sin\theta_v = |u_1||u_2|sin\theta_u$ as shown in figure 18.1(b). In an orthogonal basis we can simply choose the smallest vector in the basis.

Our transformation at each stage is analagous to euclid: we take two vectors $u, v \to u, u - v$ and repeat. In two dimensions our algorithm looks like following.

```
while (|v2| > |v2 − v1|) or (|v2| > |v2 + v1|) do
    v2 := min(|v2 − v1|, |v2 + v1|)
    if |v2| < |v1| then swap(v1, v2)
done
```

The important thing to verify is that the lattice does not change with the new vector. This is because we can recover the old $v_2$ by the inverse of the vector we chose in the min. Also we not that this is a vector in

the old lattice since we have chosen the weights.

This section was just for intuition. We have not shown that this is poly-time or exact. These are proofs are done by case analaysis and are not provided.

### 18.3.2 An approximation algorithm

In higher dimensions, we will use an approximation algorithm so that we can prove a polynomial running time. We will begin in dimension 2 to get most of the key ideas. Also we will be using the $l_2$ norm which has the property that its distances are preserved under rotation of bases.

Given $\{v_1, v_2\}$ we write them in the following normal form. $\begin{bmatrix} v_{11} & 0 \\ v_{21} & v_{22} \end{bmatrix}$ For convenience of notation, we will talk about the vector $\vec{v_{22}} = (0, v_{22})$ and the vector $\vec{v_{21}} = (v_{21}, 0)$. Note the determinant of this matrix is $v_{11} * v_{22}$. This also proves our parallelpiped result. It only takes poly time to get this form because it requires multiplying by a rotation matrix. Both the derivation and the multiplication are polytime.

**Definition 18.2.** We say a basis is *weakly reduced* if we can write it as $\begin{bmatrix} v_{11} & 0 \\ \mu_{21} v_{11} & v_{22} \end{bmatrix}$ with $|\mu_{21}| < \frac{1}{2}$.

The conditions for the exact algorithm are

1. the basis is weakly reduced
2. $||v_{11}|| \leq ||\vec{v_{21}} + \vec{v_{22}}||$

We will relax the latter condition for our approximation.

1. the basis is weakly reduced
2. $||v_{11}|| \leq \frac{2}{\sqrt{3}} ||\vec{v_{21}} + \vec{v_{22}}||$

We must now show that the vector we find by this process is *almost* shortest and the resulting algorithm runs in poly time.

**Lemma 18.7.** *Algorithm produces a basis such that* $||v_{11}|| \leq \sqrt{2}\, min\{||v_{11}||, ||v_{22}||\}$

*Proof.*

$$||v_{11}||^2 \leq \frac{4}{3} ||\vec{v_{21}} + \vec{v_{22}}||^2$$

$$= \frac{4}{3}(||v_{21}||^2 + ||v_{22}||^2) \text{ (perpendicular)}$$

$$= \frac{4}{3}(|\mu_{21}^2|||v_{11}||^2 + ||v_{22}||^2) \Rightarrow$$

$$\frac{2}{3}||v_{11}||^2 \leq \frac{4}{3}||v_{22}||^2$$

$$\Rightarrow ||v_{11}||^2 \leq 2||v_{22}||^2$$

This shows it is a $\sqrt{2}$ approximation. □

**Lemma 18.8.** *The vector found by this procedure* $v_{11}$ *has* $||v_{11}|| \leq \sqrt{2}||$*shortest vector*$||$.

*Proof.* The shortest vector must be representable in the basis. Let it equal $m_1 v_1 + m_2 v_2$.

**case 1:** $m_2 = 0$ This means it's only shortest when $m_1 = \pm 1$ i.e. is equal to $||v_1||$.

**case 2:** $m_2 \neq 0$ This means $|m_2| \geq 1$. Evaluating its length:

$$||\text{shortest vector}|| = ||m_1 \vec{v_1} 1 + m_2(\vec{v_{21}} + \vec{v_{22}})||$$
$$= ||\underline{m_1 v_{11} + m_2 \mu_{21} v_{11}} + \underline{m_2 v_{22}}||$$

Notice that the underlined portions are perpendicular. We can conlcude $||SV||^2 \geq ||m_2 v_{22}||^2 \Rightarrow ||SV||^2 \geq ||v_{22}||^2 \Rightarrow \sqrt{2}||SV|| \geq ||v_{11}||$, by previous lemma. □

*Remark.* Can choose other factors $\in (1, \frac{\sqrt{3}}{2})$.

Now we must show it is polytime. To do so we are going to define a potential function, $\theta = ||v_{11}||$ at each step of the iteration. $||v_{11}||$ is a non-zero integer vector $\Rightarrow \geq 1$. At each swap, we get that $|v_{11}| > \frac{2}{\sqrt{3}}|v_2|$ so it is cutdown by $\frac{\sqrt{3}}{2}$ at each interval therefore it executes in $log_{\frac{\sqrt{3}}{2}}(v_{11})$

Now we have the basic tools to generalize this algorithm to higher dimensions.