Master's Thesis

Lipschitz Constants of Functions of Neural Networks

Sunjoong Kim

Department of Mathematics

Graduate School

Korea University

February 2024

Lipschitz Constants of Neural Networks

by
Sunjoong Kim

---

under the supervision of Professor Seungsang Oh

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Mathematics

Department of Mathematics

Graduate School

Korea University

October 2023

The thesis of Sunjoong Kim has been approved
by the thesis committee in partial fulfillment
of the requirements for the degree of
Master of Mathematics

December 2023

Committee Chair: Name

Committee Member: Name

Committee Member: Name

Lipschitz Constants of Neural Networks

by Sunjoong Kim

Department of Mathematics

under the supervision of Professor Seungsang Oh

Abstract

In machine learning and deep learning, making an algorithm stable or robust is just as important as making it perform well. One of the indicator that measures the sensitiveness or the robustness of an algorithm is the Lipschitz constant. This paper discusses several ways to calculate the Lipschitz constant of basic machine learning algorithms. One of the main part of the perceptron and MLP is the linear function, whose Lipschitz constant coincide with the operator norm of the corresponding matrix. Unlike the linear function, evaluating the optimal Lipschitz constant of many algorithms is difficult or infeasible even for the two layered MLP. Instead of calculating the exact constant, there is a systematic way called *AutoLip* to find an upper bound for the constant and we apply it to MLP and CNN. For example, the convolution layer can be thought of as a matrix multiplication by a long matrix and it may require several tools to make it simple.

Keywords: Lipschitz constant, Operator norm, Rademacher's theorem, AutoLip, Power method, Rayleigh quotient

인공신경망의 립시츠 상수

김 선 중
수 학 과
지 도 교 수: 오 승 상

국문 초록

머신러닝과 딥러닝에서 안정적인 알고리즘을 만드는 것은 그 알고리즘이 좋은 성능을 내는 것 만큼이나 중요하다. 알고리즘의 안정성과 관련되어 있는 수학적인 개념 중 하나는 립시츠 상수이다. 이 논문에서는 기본적인 머신러닝 알고리즘의 립시츠 상수를 계산하는 다양한 방법에 대해 논의한다. 가장 기본적인 구조인 퍼셉트론은 선형함수의 구조를 가지고 있고, 선형함수의 립시츠 상수는 그 선형함수에 대응되는 행렬에 대한 작용소 놈의 값과 일치한다. 하지만, 선형함수에서 조금만 알고리즘이 복잡해져도 립시츠 상수를 계산하는 것은 어려워진다. 예를들어, 레이어가 두 개인 다층퍼셉트론의 경우만 해도 정확하게 립시츠 상수를 계산하는 것이 거의 불가능하다 (NP-hard). 그래서 이 논문에서는, 다양한 머신러닝 알고리즘의 정확한 립시츠 상수 값을 계산하는 대신 립시츠 상수의 상한(upper bound)을 계산하는 체계적인 알고리즘 (AutoLip)을 소개하고, 이것을 다층퍼셉트론과 합성곱신경망에 적용해본다. 합성곱 레이어는, 특정한 형태의 행렬의 행렬곱으로 이해할 수 있고, 따라서 작용소 놈을 계산하면 립시츠 상수가 얻어진다. 이때, 효율적인 계산을 위해 누승법이나 레일리몫과 같은 방법이 사용될 수 있다.

중심어 : 립시츠 상수, 작용소 놈, Rademacher 정리, AutoLip, 누승법, 레일리 몫

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

Deep neural networks have made many advances including computer vision, language modeling, machine translation and text and picture generating. Still, one of the difficulties in applying deep learning algorithms to reality is that the algorithm often lacks its robustness. As an example, Szegedy et al. have found that a small perturbation of an input for true image may cause the classifier to misclasssify the image as false image [2].

To overcome the instabiltiy of training and to enhance robustness of generating models such as GANs, M. Arjovsky et al. proposed Wasserstein distance between distributions and restrict their attention to 1-Lipschitz function to the critic [3], [4]. As this example suggest, the Lipschitz constant can be a good metric to access the robustness of the algorithm.

The Lipschitz constant of a function measures the sensitiveness or the robustness, or the rate of changes of the function. In chapter 2, we define the optimal Lipschitz constant of the functions between euclidean spaces and explore the computation of this optimal constant for various functions including linear maps, affine maps and compositions of functions. However, if the function becomes more complex, it is difficult or almost impossible to calculate the optimal constant. So, we propose algorithms to estimate the constant, using the Rademacher's theorem.

Chapter 2. Lipschitz Constants of Functions Between Euclidean Spaces

2.1 Lipschitz Constants

For vectors $x = [x_1 \ \cdots \ x_n]^T$ and $y = [y_1 \ \cdots \ y_m]^T$ in Euclidean spaces $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively, $||x||$ and $||y||$ are the usual standard norms of x and y defined by

$$||x|| = \sqrt{{x_1}^2 + \cdots + {x_n}^2}, \quad ||y|| = \sqrt{{y_1}^2 + \cdots + {y_m}^2}. \tag{2.1.1}$$

Definition 1. *A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is called* Lipschitz continuous *if there is a nonnegative real number c such that*

$$||f(x) - f(y)|| \leq c||x - y|| \qquad (x, y \in \mathbb{R}^n) \tag{2.1.2}$$

The infimum of c which satisfies (2.1.2) *is called the* Lipschitz constant *of f and is denoted by $Lip(f)$.*

This definition can also be extended to functions from a metric space to another metric space if we substitute the norms with the distances. It is useful to know that there

is an equivalent definition ;

$$\mathrm{Lip}(f) = \sup_{x \neq y} \frac{||f(x) - f(y)||}{||x - y||} \tag{2.1.3}$$

To show the equivalence of the two definitions, let

$$\begin{aligned} L_1 &= \inf\{c \geq 0 : ||f(x) - f(y)|| \leq c||x - y||, \quad x, y \in \mathbb{R}^n\} \\ L_2 &= \sup\left\{ \frac{||f(x) - f(y)||}{||x - y||} : x, y \in \mathbb{R}^n, x \neq y \right\}. \end{aligned} \tag{2.1.4}$$

Let $\mathscr{L}_1$ and $\mathscr{L}_2$ be two the sets in the definition of L_1 and L_2, respectively. Then, $\mathscr{L}_2 \subset \mathscr{L}_1$ since

$$\mathscr{L}_2 = \{c \geq 0 : ||f(x) - f(y)|| = c||x - y||, \quad x, y \in \mathbb{R}^n, x \neq y\},$$

and it follows that $L_1 \leq L_2$. Pick $c \in \mathscr{L}_1$. Then, $||f(x) - f(y)|| \leq c||x - y||$ for all x and y in $\mathbb{R}^n$. Assuming $x \neq y$, we have

$$\frac{||f(x) - f(y)||}{||x - y||} \leq c.$$

Taking supremum for all x and y ($x \neq y$), we have $L_2 \leq c$. and taking infimum for all $c \leq \mathscr{L}_1$ yields $L_2 \leq L_1$. Thus the two definitions are indeed equivalent.

The optimal Lipschitz constant $\mathrm{Lip}(f)$ is not only the infimum, but also the minimum of L satisfying (2.1.2). To show this, it suffices to prove that $\mathrm{Lip}(f) \in \mathscr{L}_1$, or that $c = \mathrm{Lip}(f)$ satisfies the condition. Let x, $y \in \mathbb{R}^n$. If $x = y$, then (2.1.2) holds trivially. Assuming $x \neq y$, we have

$$\frac{||f(x) - f(y)||}{||x - y||} \leq \mathrm{Lip}(f)$$

because of the second definition (2.1.3) of $\mathrm{Lip}(f)$. Multiplying both sides by $||x - y||$, we

can conclude that $c = \mathrm{Lip}(f)$ satisfies the condition (2.1.2).

Note also that $\mathrm{Lip}(f) = 0$ if and only if f is a constant function. Suppose that f is a constant function. Then, the condition (2.1.2) holds vacuously and $\mathscr{L}_1 = [0, \infty)$; $\mathrm{Lip}(f) = 0$. Suppose, on the contrary, that $\mathrm{Lip}(f) = 0$. Pick $x, y \in \mathbb{R}^n$. Since $0 \in \mathscr{L}_1$, we have $||f(x) - f(y)|| = 0$. Thus, $f(x) = f(y)$ and f is a constant function.

2.2 Real-valued Functions of a Real Variable

For univariate real function $f : \mathbb{R} \to \mathbb{R}$, the above condition (2.1.2) is equivalent to saying that the average rate of change is upper bounded by c;

$$\left| \frac{f(x) - f(y)}{x - y} \right| \leq c. \tag{2.2.5}$$

If f is diffrentiable everywhere in the domain, the mean value theorem guarantees that this is equivalent to

$$|f'(x)| \leq c \tag{2.2.6}$$

for every $x \in \mathbb{R}$. Moreover, $\mathrm{Lip}(f)$ is the least nonnegative number c satisfying the condition (2.2.6) for all x and we can express $\mathrm{Lip}(f)$ in its exact form ;

$$\mathrm{Lip}(f) = \sup_{x \in \mathbb{R}} |f'(x)| \tag{2.2.7}$$

In this sense, we can easily evaluate the optimal Lipschitz constant of the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2.8}$$

and the hyperbolic tangent function

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{2.2.9}$$

Since $0 < \sigma(x) < 1$ and $\sigma'(x) = \sigma(x)\,(1 - \sigma(x))$, we have $\sigma'(x) < \frac{1}{4}$ so that

$$\mathrm{Lip}(\sigma) = \frac{1}{4}. \tag{2.2.10}$$

Since $\tanh(x) = 2\sigma(2x) - 1$, we have

$$\tanh'(x) = 4\sigma'(2x) < 1$$

and

$$\mathrm{Lip}(\tanh) = 1. \tag{2.2.11}$$

The ReLU function

$$\mathrm{ReLU}(x) = \max\{0, x\} \tag{2.2.12}$$

is not differentiable everywhere, but we can use (2.2.5) to conclude

$$\mathrm{Lip}(\mathrm{ReLU}) = 1. \tag{2.2.13}$$

Here is a list of the optimal Lipschitz constants of univariate activation functions, frequently used in machine learning and deep learning. (**Table** 2.1)

In the definition of $\mathrm{Lip}(f)$ in (2.1.3), we can't replace the supremum by the maximum. Consider a function ; $f(x) = \ln(1 + e^x)$. This function is called the softplus activation function and is an antiderivative of the sigmoid function. It is a monotonically

Table 2.1: The Lipschitz constants of univariate activation functions.

Activation Functions	Formula	Lip(f)
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	$\frac{1}{4}$
Hyperbolic tangent	$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	1
Rectified Linear Unit	$\mathrm{ReLU}(x) = \max\{0, x\}$	1
Leaky ReLU	$\mathrm{LReLU}(x) = \max\{\alpha x, x\}$	1
Exponential Linear Unit	$\mathrm{ELU}(x) = \begin{cases} x & (x \geq 0) \\ \alpha(e^{x}-1) & (x < 0) \end{cases}$	1
Softplus	$f(x) = \frac{1}{\beta}\log(1+e^{\beta x})$	1
Gaussian	$g(x) = e^{-x^2}$	$\sqrt{\frac{2}{e}}$

increasing function whose derivative is also increasing, with an assymptote $y = x$. Thus, $\mathrm{Lip}(f) = 1$, but no x and y satisfy the equality $\frac{f(x)-f(y)}{x-y} = 1$.

2.3 Linear Functions

2.3.1 Operator Norms $||W||$

Consider a linear function $f(x) = Wx$ for some $m \times n$ matrix W. Because of the linearity, the condition (2.1.2) reduces to

$$||Wx|| \leq c||x||. \qquad (x \in \mathbb{R}^n) \tag{2.3.14}$$

The smallest c which satisfies (2.3.14) is called *the operator norm of* W and denoted by $||W||$. Thus, the optimal Lipschitz constant of f equals the operator norm of W ;

$||W|| = \mathrm{Lip}(f)$. The definitions of the forms (2.1.4) are as follows;

$$\begin{aligned} ||W|| &= \inf\{c \geq 0 \,:\, ||Wx|| \leq c||x||\} \\ &= \sup\left\{ \frac{||Wx||}{||x||} \,:\, x \in \mathbb{R}^n, x \neq 0 \right\} \end{aligned} \tag{2.3.15}$$

If we modify the second definition of (2.3.15) to

$$||W|| = \sup\{||Wx|| : ||x|| = 1\}, \tag{2.3.16}$$

we can easily verify that the supremum is actually the maximum in this case. Because the set $\{x : ||x|| = 1\}$ is a compact subset of $\mathbb{R}^n$ and the map $x \mapsto ||Wx||$ is continuous, the set in (2.3.16) has its maximum.

The operator norm is indeed a norm of a vector space ;

Proposition 1. *The operator $||\cdot|| : W \mapsto ||W||$ satisfies the following three properties. Thus, the set $\mathscr{M}_{m,n}(\mathbb{R})$ of all m by n real matrices is a normed vector space with respect to this norm ;*

(a) $||W|| \geq 0$ *for all* $W \in \mathscr{M}_{m,n}(\mathbb{R})$ *;* $||W|| = 0$ *if and only if* $W = 0$.

(b) $||kW|| = |k| \cdot ||W||$ *for all* $W \in \mathscr{M}_{m,n}(\mathbb{R})$ *and* $k \in \mathbb{R}$.

(c) $||W+V|| \leq ||W|| + ||V||$ *for all* $W, V \in \mathscr{M}_{m,n}(\mathbb{R})$.

For (a), that $||W|| \geq 0$ is obvious. If $||W|| = 0$, then $||Wx|| \leq 0||x|| = 0$ for all $x \in \mathbb{R}^n$. It follows that $Wx = 0$ for all $x \in \mathbb{R}^n$, by the definition of vector norm, thus $W = 0$. Suppose, on the other hand, that $W = 0$. Then $Wx = 0$ for all $x \in \mathbb{R}^n$. Thus, $c = 0$ is the minimal

value satisfying (2.3.14). To prove (b), we have

$$\begin{aligned}
||kW|| &= \inf\{c : ||kWx|| \leq c||x||\} \\
&= \inf\{c : |k| \cdot ||Wx|| \leq c||x||\} \\
&= \inf\left\{c : ||Wx|| \leq \frac{c}{|k|}||x||\right\} \\
&= \inf\{|k|b : ||Wx|| \leq b||x||\} \\
&= |k| \inf\{b : ||Wx|| \leq b||x||\} \\
&= |k| \cdot ||W||.
\end{aligned}$$

Now, consider (c). For any $x \in \mathbb{R}^n$, we have

$$\begin{aligned}
||(W+V)x|| = ||Wx+Vx|| &\leq ||Wx|| + ||Vx|| \\
&\leq ||W||\,||x|| + ||V||\,||x|| = (||W|| + ||V||)||x||
\end{aligned}$$

By the minimality of $||W+V||$, we have $||W+V|| \leq ||W|| + ||V||$.

An analogue of (c) also holds for multiplication; if $W \in \mathscr{M}_{m,n}(\mathbb{R})$ and $V \in \mathscr{M}_{n,k}(\mathbb{R})$, then

$$||WV|| \leq ||W||\,||V||. \tag{2.3.17}$$

This is because of the inequality

$$||(WV)x|| = ||W(Vx)|| \leq ||W||\,||Vx|| \leq ||W||\,||V||\,||x||$$

and the minimality of $||WV||$.

2.3.2 Evaluation of $||W||$ for square matrices

First, consider the case when W is a square matrix so that $f : \mathbb{R}^n \to \mathbb{R}^n$. Let $\lambda_1, \cdots, \lambda_n$ be (possibly repeated) eigenvalues of W and let $\tilde{\lambda} = \max\{|\lambda_i| \ : \ i = 1, \cdots, n\}$. ($\tilde{\lambda}$ is sometimes called the *dominating eigenvalue*, especially in the context of the power method : Subsection 3.4.2) Then, the following inequality holds in general ;

$$\tilde{\lambda} \leq ||W||. \tag{2.3.18}$$

For a proof, pick an eigenvalue λ_i and the corresponding eigenvector x_i which is nonzero. Since $Wx_i = \lambda_i x_i$, Then, we have $||Wx_i|| = |\lambda_i| \, ||x_i||$ and

$$|\lambda_i| = \frac{||Wx_i||}{||x_i||} \leq ||W||.$$

Taking the maximum over i, we have the desired inequality. The above inequality (2.3.18) becomes equality when W is symmetric.

Suppose that W is real symmetric. Then, W is orthogonally diagonalizable in the sense that, there exists an orthogonal matrix

$$V = [v_1 \quad \cdots \quad v_n],$$

such that

$$W = V \Lambda V^T,$$

where $\Lambda = \mathrm{diag}\{\lambda_1, \cdots, \lambda_n\}$ and λ_i is an eigenvalue of W with the corresponding eigenvector v_i. Note that $\{v_1, \cdots, v_n\}$ forms an orthonormal basis for $\mathbb{R}^n$. For any nonzero

$x \in \mathbb{R}^n$, there exist real numbers $c_1, \cdots, c_n$ such that $x = c_1 v_1 + \cdots + c_n v_n$. Since

$$\begin{aligned} Wx &= c_1 W v_1 + \cdots + c_n W v_n \\ &= c_1 \lambda_1 v_1 + \cdots + c_n \lambda_n v_n, \end{aligned}$$

we have (by the orthogonality of v_i's

$$\begin{aligned} ||Wx||^2 &= ||c_1 W v_1 + \cdots + c_n W v_n||^2 \\ &= |c_1 \lambda_1|^2 ||v_1||^2 + \cdots + |c_n \lambda_n|^2 ||v_n||^2 \\ &= {c_1}^2 {\lambda_1}^2 + \cdots + {c_n}^2 {\lambda_n}^2. \end{aligned}$$

Thus,

$$\frac{||Wx||^2}{||x||^2} = \frac{{c_1}^2 {\lambda_1}^2 + \cdots + {c_n}^2 {\lambda_n}^2}{{c_1}^2 + \cdots + {c_n}^2} \leq \tilde{\lambda}^2,$$

for which $||Wx||/||x|| \leq \tilde{\lambda}$. Therefore, we have the reverse inequality $||W|| \leq \tilde{\lambda}$.

As examples consider the following matrices W_1, W_2, W_3, W_4, W_5 defined by

$$W_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad W_2 = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}, \quad W_3 = \begin{bmatrix} 4 & 2 \\ 2 & 7 \end{bmatrix},$$

$$W_4 = \begin{bmatrix} 1 & 3 \\ 2 & 0 \end{bmatrix}, \quad W_5 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

W_1 is the identity matrix and thus it must be that $||W|| = 1$. Indeed, the characteristic polyinomial of W_1 is $(\lambda - 1)^2 = 0$, $\lambda_1 = \lambda_2 = 1$ and thus the maximum absolute value of

them is 1. The matrix W_2 stretches a given vector x twice in the x-axis direction and three times in the y-axis direction. Thus, $||W_2||$ should be 3. Indeed, $|W_2 - \lambda I| = (\lambda - 2)(\lambda - 3)$ and $\tilde{\lambda} = 3$. W_3 is the last example that is symmetric ; $|W_3 - \lambda I| = (\lambda - 3)(\lambda - 8)$ and $\tilde{\lambda} = 8$.

W_4 is not symmetric and we can't evaluate $||W_4||$ for now. Still, we have, $\lambda_1 = -2$, $\lambda_2 = 3$ and $||W_4|| \geq 3$ by (2.3.18). The reverse inequality is not valid since the eigenvectors $x_1 = [1 \ -1]^T$ and $x_2 = [3 \ 2]^T$ are not perpendicular. W_5 is not symmetric either, but we can postulate that $||W_5|| = 1$ since

$$W_5 x = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ 0 \end{bmatrix}$$

and

$$\frac{||W_5 x||}{||x||} = \frac{|x_2|}{\sqrt{{x_1}^2 + {x_2}^2}} \leq 1.$$

Since W_5 has the only eigenvector 0, the inequality (2.3.18) still holds.

Here is a short description of what we've done in this subsection.

Lemma 1. *Let $W \in \mathscr{M}_n(\mathbb{R})$ be a real symmetric matrix. Then, the operator norm $||W||$ is the maximal absolute eigenvalue of W.*

2.3.3 Evaluation of $||W||$ for rectangular matrices

Almost all matrices that we encounter in machine learning problem are neither symmetric matrices, nor are they square matrices. But, we can always calculate the Lipschitz constant or the operator norm of any rectangular matrices.

Theorem 1. *[5] Let $W \in \mathscr{M}_{m,n}(\mathbb{R})$ be an m by n real matrix. Then*

$$||W|| = \sqrt{||W^T W||}. \tag{2.3.19}$$

Note that the norm on the left hand side is the operator norm on $\mathscr{M}_{m,n}(\mathbb{R})$ and the norm on the right hands side is the operator norm on $\mathscr{M}_n(\mathbb{R})$. Note also that the matrix $W^T W$ is real symmetric in that $(W^T W)^T = W^T (W^T)^T = W^T W$ and we are always able to calculate the norm of $||W^T W||$ by making use of the lemma 1.

Before the proof of (2.3.19), we illustrate elementary properties of the operator norm of matrices. The euclidean norm $||x||$ of a vector $x \in \mathbb{R}^n$ in (2.1.1) can also be defined by means of the inner product on $\mathbb{R}^n$;

$$||x|| = \sqrt{\langle x, x \rangle}. \tag{2.3.20}$$

The inner product $\langle x, y \rangle = x^T y$ has the following properties. (we omit the proofs of them.)

$$\langle x, Wy \rangle = \langle W^T x, y \rangle,$$
$$\langle kx, y \rangle = k \langle x, y \rangle,$$
$$|\langle x, y \rangle| \leq ||x||\,||y||.$$

Now we are ready to prove (2.3.19). If $W = 0$, then the equation (2.3.19) holds triv-

ially. Suppose that $W \neq 0$. Let $x \in \mathbb{R}^n$. Then

$$\begin{aligned} ||Wx||^2 &= \langle Wx, Wx \rangle \\ &= \langle W^T W x, x \rangle \\ &\leq ||W^T W x||\, ||x|| \\ &\leq ||W^T W||\, ||x||^2, \end{aligned}$$

and thus

$$||Wx|| \leq \sqrt{||W^T W||}\, ||x||.$$

for all $x \in \mathbb{R}^n$. By the minimality of $||W||$, we have

$$||W|| \leq \sqrt{||W^T W||}$$

Squaring both sides and making use of (2.3.17) yield

$$||W||^2 = ||W^T W|| \leq ||W^T||\, ||W||. \tag{2.3.21}$$

Since $||W|| \neq 0$, we have

$$||W|| \leq ||W^T||.$$

Substituting W by its transpose, we get the reverse inequality. Therefore,

$$||W|| = ||W^T||. \tag{2.3.22}$$

By (2.3.22), the equation (2.3.21) reduces to

$$||W||^2 = ||W^T W||,$$

and this proves (2.3.19).

For example, we can evaluate the operator norm of W_4 and W_5 in the previous subsection. We have

$$W_4{}^T W_4 = \begin{bmatrix} 5 & 3 \\ 3 & 9 \end{bmatrix}, \quad W_5{}^T W_5 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

$W_4{}^T W_4$ has its eigenvalues $7 \pm \sqrt{13}$, thus $||W_4|| = \sqrt{7+\sqrt{13}}$. And $W_5{}^T W_5$ has eigenvalues 0 and 1 ; $||W_5|| = \sqrt{1} = 1$ as expected. As a rectangular matrix, we may think of

$$W_6 = \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 0 & 1 \end{bmatrix}.$$

Since

$$W_6{}^T W_6 = \begin{bmatrix} 5 & 2 \\ 2 & 2 \end{bmatrix},$$

$\lambda = 1, 6$ and $||W_6|| = \sqrt{6}$.

2.4 Affine Functions

An affine function

$$f(x) = Wx + b \tag{2.4.23}$$

has the same optimal Liptschitz constant as that of its linear part ;

$$\mathrm{Lip}(f) = ||W|| \tag{2.4.24}$$

This is because the condition (2.1.2)

$$||f(x) - f(y)|| \leq c||x - y||$$

applied to (2.4.23) becomes

$$||W(x - y)|| \leq c||x - y||.$$

2.5 Elementwise Application of Nonlinear Functions

Let $f : \mathbb{R} \to \mathbb{R}$ be any function and let $F : \mathbb{R}^m \to \mathbb{R}^m$ be applying f to each element of the input $x = [x_1 \cdots x_m]^T$:

$$F(x) = F\left(\begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}\right) = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_m) \end{bmatrix}. \tag{2.5.25}$$

If f is a Liptschitz continuous function with its optimal constant L. Then,

$$||F(x)-F(y)||^2 = \left|\left|\left|\begin{bmatrix} f(x_1)-f(y_1) \\ \vdots \\ f(x_m)-f(y_m) \end{bmatrix}\right|\right|\right|^2 = \sum_{i=1}^{m} |f(x_i)-f(y_i)|^2$$
$$\leq \sum_{i=1}^{m} L^2|x_i-y_i|^2 = L^2||x-y||^2,$$

for which

$$||F(x)-F(y)|| \leq L||x-y||.$$

Thus, $\mathrm{Lip}(F)$ is upper bounded by $L = \mathrm{Lip}(f)$;

$$\mathrm{Lip}(F) \leq \mathrm{Lip}(f). \tag{2.5.26}$$

2.6 Composites of Functions

Let $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^m \to \mathbb{R}^k$ be Lipstchitz continuous functions. Then, the composite function $g \circ f$ is also Lipstchitz continuous and the optimal constant is upper bounded by the product of those of f and g ;

$$\mathrm{Lip}(g \circ f) \leq \mathrm{Lip}(f)\mathrm{Lip}(g) \tag{2.6.27}$$

This is readily proved by the following reason. Let L and M be the optimal Lipschitz

constants for f and g respectively. Then, for all $x, y \in \mathbb{R}^n$,

$$\begin{aligned} ||(g \circ f)(x) - (g \circ f)(y)|| &= ||g(f(x)) - g(f(y))|| \\ &\leq M||f(x) - f(y)|| \\ &\leq LM||x - y||. \end{aligned}$$

By the minimality of $\mathrm{Lip}(g \circ f)$, we have (2.6.27).

Note that (2.3.17) is a specific case of (2.6.27). Here, the inequality can be strict. As an example, $||W_2||\,||W_4|| = 3 \times \sqrt{7+\sqrt{13}} > \sqrt{1+\sqrt{37}} = ||W_2 W_4||$. Note also that (2.6.27) can be generalized further, to the case when the composition involoves several functions. That is, if $f_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ are functions between euclidean spaces, with its optimal Lipschitz constant L_i, for $i = 1, 2, \cdots, p$, the composite

$$f_p \circ \cdots \circ f_1 : \mathbb{R}^{n_1} \to \mathbb{R}^{n_{p+1}}$$

is Lipschitz constinuous and

$$\mathrm{Lip}(f_p \circ \cdots \circ f_1) \leq \prod_{i=1}^{k} L_i. \tag{2.6.28}$$

Chapter 3. Lipschitz Constants of Neural Networks

Now we turn to the problem of finding the optimal Lischitz constant of various architectures frequently appearing in machine learning and deep learning.

For locally Lipschitz function, it is possible to express the Lipschitz constant of f in its exact form, as the Rademacher Theorem suggests. But, it is not always easy or feasible to find the exact value of Lipschitz constant whenever the function f is given. In fact, although the function is of simple form such as 2-layered MLP, solving the precise value of Lipschitz constant of the function is known to be NP-hard. Instead of struggling to find analytic solutions, we present a systematic algorithm for each architectuare of neural network[1].

3.1 Rademacher Theorem

Here is a theorem that can be applied to all Lipschitz functions between euclidean spaces. The Lipschitz condition that we impose is not the global one ; it only need to be locally Lipschitz. It has two conclusions : the differentiability in almost everywhere sense and the formula for the optimal constant.

Theorem 2. *[8] Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a locally Lipschitz function. Then, f is differentiable almost everywhere, and*

$$\mathrm{Lip}(f) = \sup_{x \in \mathbb{R}^n} ||D_x(f)|| \tag{3.1.1}$$

The first statement — differentiability — is usually refered to the main conclusion of this theorem. But the proof of it is so lengthy that we omit the proof here. As for one-dimensional case, many of the textbooks of analysis covered this issue. In the process, concepts like absolute continuity, Radon-Nykodym theorem, total variation are involved [7].

The second statement, or the optimal Lipschitz constant is the one that we can use for our purpose. Note first that, $D_x(f)$ is the Jacobian matrix A of f satisfying

$$\lim_{y \to x} \frac{||f(y) - f(x) - A(y-x)||}{||y-x||} = 0,$$

and that $||D_x(f)||$ is nothing but the operator norm we've defined in the previous chapter.

Note also that although the equation (3.1.1) expresses $\mathrm{Lip}(f)$ explicitly, it doesn't mean that we can actually find the value of $\mathrm{Lip}(f)$.

3.2 Upperbounding the Lipschitz constant : AutoLip

Here is an algorithm called *AutoLip* that evaluate an upperbound for the Lipschitz constant. The algorithm can be applied to any deterministic neural network (resp. stochastic neural networks or bayesian neural networks) with the computation graph, which is a composite of elementary functions.

The only difficulty in the Rademacher theorem was the expression supremum ; it

is not easy or sometimes infeasible to evaluate the exact value of the supremum or the maximum of a function. But, it is possible, in most case, to find the maximum in the relatively simple function. In other words, the supremum for the Jacobian matrix can be calcuated for most elementary functions.

For a function $f : \mathbb{R}^n \to \mathbb{R}^m$, we've used the notation for the Jacobian matrix of f as

$$D_x(f)$$

in the previous section. In this section, we occasionally use

$$\frac{\partial y}{\partial x}$$

to represent the same matrix, provided that $y = f(x)$. In this context, $\left|\left|\frac{\partial y}{\partial x}\right|\right|$ represents the operator norm for the matrix. Furthermore, if y is a function of a univariate variable x, usual notation for the derivative is $\frac{dy}{dx}$. But, in order to unify the notation, we stick to use the partial derivative notation $\frac{\partial y}{\partial x}$.

3.2.1 An exmaple for univariate case

Consider, for example, a univariate function f_ω with a parameter ω defined by

$$f_\omega(x) = \frac{1}{1+e^{-x}} + |2x + \omega \cos x| .$$

In order to construct a systematic way to estimate the constant and to make use of automatic differentiation in implementing the backpropagation by Tensorflow or Pytorch, we think of a computation graph for f_ω. (Figure 3.1)

We can bound the Lipschitz constant of f_ω from above, by upperbounding the partial

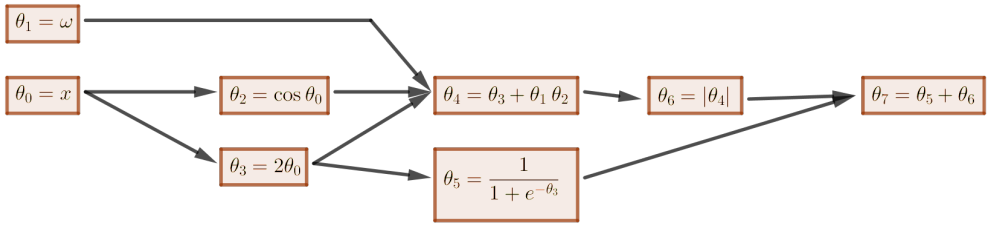

Figure 3.1: A computation graph for the function $f_\omega(x) = \frac{1}{1+e^{-2x}} + |2x + \omega\cos x|$.

derivatives of θ_1, θ_2, $\cdots$, θ_7 consecutively where $\theta_7 = f_\omega(x)$. Since $\theta_1 = g_1(\theta_0) = \omega$, we have $\frac{\partial\theta_1}{\partial x} = 0$. The next parameter θ_2 is a function of one variable θ_0, but we regard θ_2 as a function of two variables ; $\theta_2 = g_2(\theta_0, \theta_1) = \cos\theta_0$. It is legal to apply the chain rule to get

$$\begin{aligned}\frac{\partial\theta_2}{\partial x} &= \frac{\partial\theta_2}{\partial\theta_0}\frac{\partial\theta_0}{\partial x} + \frac{\partial\theta_2}{\partial\theta_1}\frac{\partial\theta_1}{\partial x} \\ &= (-\sin\theta_0)\times 1 + 0\times 0 \\ &= -\sin x.\end{aligned}$$

Then, the Lipschitz constant $\mathrm{Lip}(\theta_2)$ of θ_2 is

$$\left|\left|\frac{\partial\theta_2}{\partial x}\right|\right| = \sup_x |-\sin x| = 1.$$

Again, since $\theta_3 = g_3(\theta_0, \theta_1, \theta_2) = 2\theta_0$, we have

$$\begin{aligned}\frac{\partial\theta_3}{\partial x} &= \frac{\partial\theta_3}{\partial\theta_0}\frac{\partial\theta_0}{\partial x} + \frac{\partial\theta_3}{\partial\theta_1}\frac{\partial\theta_1}{\partial x} + \frac{\partial\theta_3}{\partial\theta_2}\frac{\partial\theta_2}{\partial x} \\ &= 2\times 1 + 0\times 0 + 0\times(-\sin x) = 2,\end{aligned}$$

and thus $\left|\left|\frac{\partial \theta_2}{\partial x}\right|\right| = 2$. Note that the value of partial derivatives evaluated above, such as $\frac{\partial \theta_1}{\partial x}$ and $\frac{\partial \theta_2}{\partial x}$, can be applied to calcuate $\frac{\partial \theta_3}{\partial x}$. To represent that $\left|\left|\frac{\partial \theta_3}{\partial x}\right|\right|$ is upper bounded by 2, we write $L_3 = 2$. Likewise, let $L_1 = 0$ and $L_0 = 1$.

For the next parameter $\theta_4 = g_4(\theta_0, \theta_1, \theta_2, \theta_3) = \theta_3 + \theta_1\theta_2$, we have

$$\begin{aligned}\frac{\partial \theta_4}{\partial x} &= \frac{\partial \theta_4}{\partial \theta_0}\frac{\partial \theta_0}{\partial x} + \frac{\partial \theta_4}{\partial \theta_1}\frac{\partial \theta_1}{\partial x} + \frac{\partial \theta_4}{\partial \theta_2}\frac{\partial \theta_2}{\partial x} + \frac{\partial \theta_4}{\partial \theta_3}\frac{\partial \theta_3}{\partial x} \\ &= 0 \times 1 + \theta_2 \times 0 + \theta_1 \times (-\sin x) + 1 \times 2\end{aligned}$$

and

$$\begin{aligned}\left|\left|\frac{\partial \theta_4}{\partial x}\right|\right| &\leq |\omega| \cdot \left|\left|\frac{\partial \theta_2}{\partial x}\right|\right| + 1 \times \left|\left|\frac{\partial \theta_3}{\partial x}\right|\right| \\ &\leq |\omega| \cdot L_2 + L_3 = |\omega| + 2.\end{aligned}$$

Thus, $L_4 = |\omega| + 2$.

In similar fashions, $\theta_5 = g_5(\theta_0, \theta_1, \cdots, \theta_4) = \frac{1}{1+e^{-\theta_3}}$, $\theta_6 = g_6(\theta_0, \theta_1, \cdots, \theta_5) = |\theta_4|$ and $\theta_7 = g_7(\theta_0, \theta_1, \cdots, \theta_6) = \theta_5 + \theta_6$ have

$$\begin{aligned}
&\frac{\partial \theta_5}{\partial x} = \frac{\partial \theta_5}{\partial \theta_3}\frac{\partial \theta_3}{\partial x} && \left|\left|\frac{\partial \theta_5}{\partial x}\right|\right| \leq \frac{1}{4} \times 2 = \frac{1}{2} \\
&\frac{\partial \theta_6}{\partial x} = \frac{\partial \theta_6}{\partial \theta_4}\frac{\partial \theta_4}{\partial x} && \left|\left|\frac{\partial \theta_6}{\partial x}\right|\right| \leq \left|\left|\frac{\partial \theta_6}{\partial \theta_4}\right|\right| \left|\left|\frac{\partial \theta_6}{\partial \theta_4}\right|\right| = |\omega| + 2 \\
&\frac{\partial \theta_7}{\partial x} = \frac{\partial \theta_7}{\partial \theta_5}\frac{\partial \theta_5}{\partial x} + \frac{\partial \theta_7}{\partial \theta_6}\frac{\partial \theta_6}{\partial x} && \left|\left|\frac{\partial \theta_7}{\partial x}\right|\right| \leq \left|\left|\frac{\partial \theta_5}{\partial x}\right|\right| + \left|\left|\frac{\partial \theta_6}{\partial x}\right|\right| \leq |\omega| + \frac{5}{2}.
\end{aligned}$$

Thus, the Lipschitz constant for the function f_ω is upper bounded by $\omega + \frac{5}{2}$.

3.2.2 AutoLip

Consider $f : \mathbb{R}^n \to \mathbb{R}^m$, a composite function of elementary functions g_0, g_1, g_2, $\cdots$, g_K, so that

$$f = g_K \circ \cdots \circ g_2 \circ g_1 \circ g_0.$$

For every $x \in \mathbb{R}^n$, define the intermediate variables $\theta_0 = g_0(x)$, $\theta_1 = g_1(g_0(x))$, $\cdots$, $\theta_K = g_K(g_{K-1}(\cdots g_1(g_0(x)))) = f(x)$. For a notational purpose, let g_0 be the identity function and think of g_k as a function of θ_0, θ_1, $\cdots$, θ_{k-1} for $1 \leq k \leq K$ and write

$$\begin{aligned}
\theta_1 &= g_1(\theta_0) \\
\theta_2 &= g_2(\theta_0, \theta_1) \\
&\vdots \\
\theta_K &= g_K(\theta_0, \theta_1, \cdots, \theta_{K-1}).
\end{aligned}$$

An upper bound L_k for the lipschitz constant $\sup_x \left|\left| \frac{\partial \theta_k}{\partial x} \right|\right|$ of θ_k, can be evaulated inductively, where L_K is an upper bound for $\mathrm{Lip}(f)$.

Let $L_0 = 1$. Suppose that suppose that k is an integer such that $1 \leq k \leq K$ and we have an upper bound L_i of the Lipschitz constant

$$\sup_x \left|\left| \frac{\partial \theta_i}{\partial x} \right|\right| \leq L_i$$

for $1 \leq i \leq k-1$. Since

$$\begin{aligned}\frac{\partial \theta_k}{\partial x} &= \frac{\partial \theta_k}{\partial \theta_0}\frac{\partial \theta_0}{\partial x} + \frac{\partial \theta_k}{\partial \theta_1}\frac{\partial \theta_1}{\partial x} + \cdots + \frac{\partial \theta_k}{\partial \theta_{k-1}}\frac{\partial \theta_{k-1}}{\partial x} \\ &= \sum_{i=0}^{k-1} \frac{\partial \theta_k}{\partial \theta_i}\frac{\partial \theta_i}{\partial x},\end{aligned}$$

it follows that

$$\left|\left|\frac{\partial \theta_k}{\partial x}\right|\right| \leq \sum_{i=0}^{k-1} \left|\left|\frac{\partial \theta_k}{\partial \theta_i}\right|\right| \left|\left|\frac{\partial \theta_i}{\partial x}\right|\right|$$

and that

$$\sup_x \left|\left|\frac{\partial \theta_k}{\partial x}\right|\right| \leq \sum_{i=0}^{k-1} \sup \left|\left|\frac{\partial \theta_k}{\partial \theta_i}\right|\right| \cdot L_i.$$

Let L_k be the right hand side of the above inequality and proceed for $k+1$. After K iterations, we have an upper bound L_K of the lipschitz constant for $\theta_K = f(x)$. (**Algorithm** 1)

input : the function $f = g_K \circ \cdots \circ g_0$, where $\theta_k = (g_k \circ \cdots \circ g_0)(x)$ for $0 \leq k \leq K$.
output: An upper bound L for $\mathrm{Lip}(f)$

$k \leftarrow 0$;
$L_0 \leftarrow 1$;
while $k < K$ **do**
  $L_k \leftarrow \sum_{i=0}^{k-1} \left(\sup \left|\left|\frac{\partial \theta_k}{\partial \theta_i}\right|\right|\right) L_i$;
  $k \leftarrow k+1$;
end
$L \leftarrow L_K$

Algorithm 1: AutoLip

3.3 Multi-Layer Perceptrons

Consider a single layer perceptron (Figure 3.2). Denote the single layer perceptron by $f_\omega : \mathbb{R}^3 \to \mathbb{R}^4$, which is defined as

$$f_\omega(x) = \mathrm{ReLU}(Wx + b),$$

where ω stands for the set of parameters : $\omega = (W, b)$. With the computation graph illustrated in Figure 3.3, we have

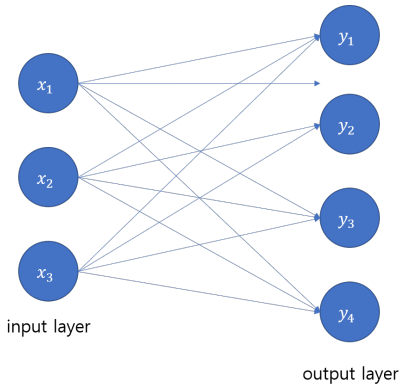

Figure 3.2: A single layer perceptron of input dimension 3 and output dimension 4 with weight matrix W and the bias b.

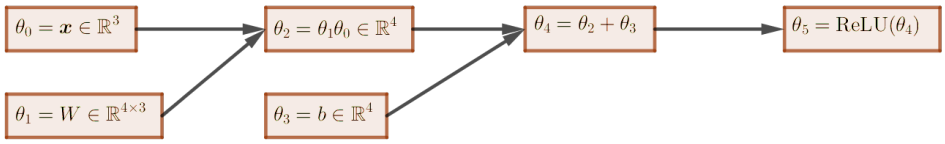

Figure 3.3: A computation graph for a single layer perceptron

$$
\begin{aligned}
&k=0;\ L_0=1\\
&k=1;\ L_1=\sup\left|\left|\frac{\partial\theta_1}{\partial\theta_0}\right|\right|\times L_0=0\times 1=0\\
&k=2;\ L_2=\sup\left|\left|\frac{\partial\theta_2}{\partial\theta_0}\right|\right|\times L_0+\sup\left|\left|\frac{\partial\theta_2}{\partial\theta_1}\right|\right|\times L_1\\
&\qquad\quad =||W||\times 1+||x||\times 0=||W||\\
&k=3;\ L_3=\sup\left|\left|\frac{\partial\theta_3}{\partial\theta_0}\right|\right|\times L_0+\sup\left|\left|\frac{\partial\theta_3}{\partial\theta_1}\right|\right|\times L_1+\sup\left|\left|\frac{\partial\theta_3}{\partial\theta_2}\right|\right|\times L_2=0\\
&k=4;\ L_4=\sup\left|\left|\frac{\partial\theta_4}{\partial\theta_0}\right|\right|\times L_0+\sup\left|\left|\frac{\partial\theta_4}{\partial\theta_1}\right|\right|\times L_1+\sup\left|\left|\frac{\partial\theta_4}{\partial\theta_2}\right|\right|\times L_2+\sup\left|\left|\frac{\partial\theta_4}{\partial\theta_3}\right|\right|\times L_3\\
&\qquad\quad =0\times 1+0\times 0+1\times||W||+1\times 0=||W||\\
&k=5;\ L_5=\sup\left|\left|\frac{\partial\theta_5}{\partial\theta_0}\right|\right|\times L_0+\sup\left|\left|\frac{\partial\theta_5}{\partial\theta_1}\right|\right|\times L_1+\sup\left|\left|\frac{\partial\theta_5}{\partial\theta_2}\right|\right|\times L_2+\sup\left|\left|\frac{\partial\theta_5}{\partial\theta_3}\right|\right|\times L_3+\sup\left|\left|\frac{\partial\theta_5}{\partial\theta_4}\right|\right|\\
&\qquad\quad =0\times 1+0\times 0+0\times||W||+0\times 0+1\times||W||=||W||.
\end{aligned}
$$

The resulting upper bound for the Lipschitz constant is $||W||$. Note that we've used the fact that $\mathrm{Lip}(\mathrm{ReLU})=1$.

Let f be a multi layer perceptron of N layers, with 1-Lipschitz activation functions (like ReLU, Leaky ReLU, tanh, or SoftPlus). Then, we can think of f as a composite of $f_1, f_2, \cdots, f_N$ where each f_i is a single layer perceptron described in the above paragraph, for $1\leq n\leq N$. Then,

$$\mathrm{Lip}(f)\leq\prod_{i=1}^{n}\mathrm{Lip}(f_n)\leq\prod_{i=1}^{n}||W_n||,$$

where each W_n is a weight matrix in the nth layer.

3.4 Convolutional Neural Networks

A vanilla convolutional neural network is a composition of convolutional layers, max pooling layers and linear layers. VGG-16, for example, consists of 13 convolutional layers, 5 maxpooling layers, 3 fully connected layers and a softmax function in the output layer [16]. While linear layers are what we've discussed in the previous section, we need to evaluate the Lipschitz constant of convolutional layers and max pooling layers.

3.4.1 Convolutional layers

Consider a convolutional layer of 5×5 input array and 3×3 filters. To make it simple, suppose that it has only one channel and that padding option is set to `valid` so that the output array is of 3×3. Denote the input array, the filter and the output array by X, F and Y. Instead of viewing X and Y as matrices(e.g. $X = [x_{ij}]_{5\times5}$, $Y = [y_{ij}]_{3\times3}$), let X and Y be column vectors of dimension 25 and 9, respectively, so that

$$X = \begin{bmatrix} \\ x_{11} \\ \\ x_{12} \\ \\ \vdots \\ \\ x_{55} \end{bmatrix}, \quad Y = \begin{bmatrix} \\ y_{11} \\ \\ y_{12} \\ \\ \vdots \\ \\ y_{33} \end{bmatrix}.$$

Then, we can think of the convolutional layer as left multiplication of a matrix F (Figure 3.4).

The Lipschitz constant for the convolution layer is, thus, the operator norm of the

matrix F and we can use the Theorem 1.

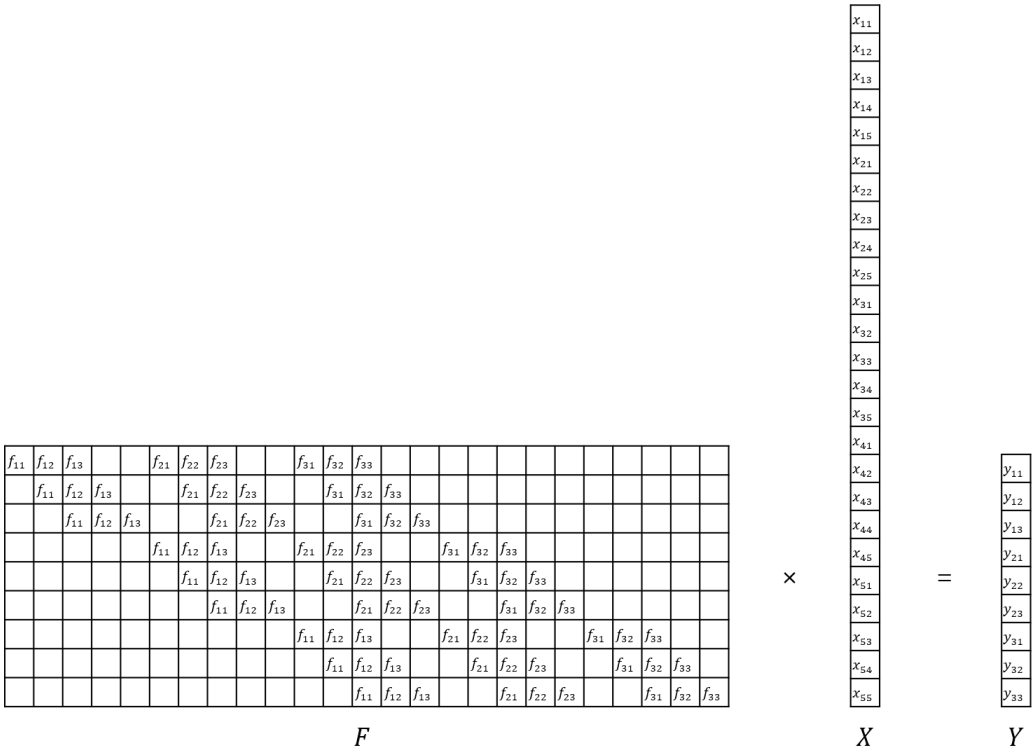

Figure 3.4: A convolutional layer as a matrix multiplication

But, it is complicated and time consuming to evaluate the Lipschitz constant directly. Instead of applying the Theorem 1 and evaluate $||F|| = \sqrt{||F^T F||}$ rigorously, we can use numerical method called *power method*.

3.4.2 Power method

Supposet that $A \in \mathbb{R}^{n \times n}$ is an orthogonally diagonalizable matrix, or that A has distinct eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_n$. This is usually the case, since the probability that the characteristic equation happen to have repeated root is 0.

Then, we have not only the distinct eigenvalues $\lambda_1, \cdots, \lambda_n$, but also the linearly inde-

pendent eigenvectors $v_1, \cdots, v_n$. Without loss of generality, let $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$, so that λ_1 is the *dominating eigenvalue* ; $|\lambda_1| = \max_i |\lambda_i|$.

Let x be any nonzero vector in $\mathbb{R}^n$. There exist $c_1, \cdots, c_n$ such that

$$x = c_1 v_1 + \cdots + c_n v_n.$$

Then,

$$\begin{aligned} Ax &= A(c_1 v_1 + \cdots + c_n v_n) \\ &= c_1 A v_1 + \cdots + c_n A v_n \\ &= c_1 \lambda_1 v_1 + \cdots + c_n \lambda_n v_n. \end{aligned}$$

Moreover,

$$\begin{aligned} A^2 x &= A(c_1 \lambda_1 v_1 + \cdots + c_n \lambda_n v_n) \\ &= c_1 \lambda_1 A v_1 + \cdots + c_n \lambda_n A v_n \\ &= c_1 {\lambda_1}^2 v_1 + \cdots + c_n {\lambda_n}^2 v_n. \end{aligned}$$

In the similar fashion, we have, for any positive integer k,

$$A^k x = c_1 {\lambda_1}^k v_1 + \cdots + c_n {\lambda_n}^k v_n.$$

Among n terms $c_1 {\lambda_1}^k v_1, \cdots, c_n {\lambda_n}^k v_n$, the first term is the dominating term, since

$$A^k x = {\lambda_1}^k \left(c_1 v_1 + c_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k v_2 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1} \right)^k v_n \right)$$

and $\left(\frac{\lambda_i}{\lambda_1}\right)^k \to 0$ exponentially as $k \to \infty$, for $i = 2, \cdots, n$. In short, for sufficient large integer k, we have

$$A^k x \approx c_1 {\lambda_1}^k v_1.$$

Furthermore, we can approximately think of $A^k x$ as the eigenvector, corresponding eigenvalue of which is the dominating one. Now, we can use the following lemma, stating that if we are given an eigenvector, we can evaluate the the corresponding eigenvalue.

Lemma 2. ***(Rayleigh Quotient)*** *If x is an eigenvector of a matrix A, then its corresponding eigenvalue is given by*

$$\lambda = \frac{\langle Ax, x \rangle}{\langle x, x \rangle}.$$

Here, $\langle \cdot, \cdot \rangle$ is the inner product as defined in Subsection 2.3.3.

Therefore, given a square orthogonally diagonalizable matrix A, we can approximate the *dominating eigenvector* x_1, by computing repeatedly $A^k x$ for any nonzero vector x, and also the corresponding *dominating eigenvalue* λ_1 by the above lemma, which converges to $||A||$. For rectangular matrix f, for example, as in Subsection 3.4.1, approximating algorithm for $||F||$ is illustrated in **Algorithm** 2.

input : A rectangular matrix $F \in \mathbb{R}^{n \times m}$.
output: Approximation for $||F||$

1 $A \leftarrow F^T F$;
2 Set an integer k sufficiently large enough;
3 Set a nonzero vector $x \in \mathbb{R}^m$;
4 $x_k \leftarrow A^k x$;
5 $\tilde{\lambda} \leftarrow \frac{{x_k}^T A x_k}{{x_k}^T x_k}$;
6 $||F|| \leftarrow \sqrt{\tilde{\lambda}}$

Algorithm 2: Approximating $||F||$ using the power method

3.4.3 Max pooling layers

The Lipschitz constant of the max pooling component is 1, by the following obvious reason.

Consider a usual 2×2 maxpooling, applied to 2d-array X. Suppose that X consists $4N$ numbers $x_1, x_2, x_3, x_4, \cdots, x_{4N-3}, x_{4N-2}, x_{4N-1}, x_{4N}$ and denote X as a $4N$-dimensional vector by

$$X = [x_1 \ x_2 \ x_3 \ x_4 \ \cdots \ x_{4N-3} \ x_{4N-2} \ x_{4N-1} \ x_{4N}]^T.$$

Let $x^{(1)}, \cdots, x^{(N)}$ be four dimensional vectors defined by

$$x^{(1)} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \cdots, x^{(N)} = \begin{bmatrix} x_{4N-3} \\ x_{4N-2} \\ x_{4N-1} \\ x_{4N} \end{bmatrix}$$

and write

$$X = \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(N)} \end{bmatrix}.$$

Lemma 3. *A function* $[a_1, a_2\, a_3, a_4]^T \mapsto \max a_i$ *is 1-Lipschitz function.*

Proof. Denote $a = [a_1, a_2\, a_3, a_4]^T$ and $b = [b_1, b_2\, b_3, b_4]^T$. Let $a_{i_0} = \max a_i$. Then,

$$\max a_i - \max b_i = a_{i_0} - \max b_i \leq a_{i_0} - b_{i_0} \leq \max(a_i - b_i).$$

Similarly,

$$\max b_i - \max a_i \leq \max(a_i - b_i).$$

Thus,

$$|\max a_i - \max b_i| \leq \max(a_i - b_i),$$

and $\mathrm{Lip}(\max) = 1$. □

Now, we are ready to evaluate $\mathrm{Lip}(\mathrm{MaxPool})$. The function

$$\mathrm{MaxPool} : \mathbb{R}^{4N} \to \mathbb{R}^N$$

is defined by

$$\mathrm{MaxPool}(X) = \begin{bmatrix} \max\left(x^{(1)}\right) \\ \vdots \\ \max\left(x^{(N)}\right) \end{bmatrix}.$$

Thus,

$$\begin{aligned}
||\text{MaxPool}(X) - \text{MaxPool}(Y)||_2{}^2 &= \sum_{n=1}^{N} \left(\max\left(x^{(n)}\right) - \max\left(y^{(n)}\right) \right)^2 \\
&\leq \sum_{n=1}^{N} \left(\max\left(x^{(n)} - y^{(n)}\right) \right)^2 \\
&\leq ||\text{MaxPool}(X - Y)||_2{}^2.
\end{aligned}$$

Therefore,

$$\text{Lip}(\text{MaxPool}) = 1.$$

Chapter 4. Conclusion

For a function between metric spaces, the Lipschitz constant measures the rate of change of the function. If we conceive a neural network as a function between Euclidean spaces, the Lipschitz constant of the neural network tells us how sensitive the output is, relative to the input. One can relate the Lipschitz constant to the robustness of the algorithm. But the Lipschitz constant is not the same as the robustness of an *architecture*, in that it measures the sensitivity of an *algorithm* when the parameters are fixed. We may further study the robustness by investigating the changes of the Lipschitz constant when updating the parameters of the algorithm.

The Lipscthiz constant for relatively simple functions such as activation functions, linear functions and affine functions can be obtained either by simple calculation or the notion of matrix norm. We can express the Lipschitz constant of *any* locally Lipschitz functions by Rademacher Theorem, but evaluating the optimal constant in its exact value is difficult or not feasible. Rather, we can think of several numerical way to find an upper bound for the optimal constant either by an algorithm called AutoLip or by the Rayleigh quotient and power method. This paper doesn't include the implementation with programming code, such as Python. One may relate the calculations to the experimental results.

Reference

[1] Virmaux, A., & Scaman, K. (2018). Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31.

[2] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. *Proceedings of the International Conference on Learning Representations (ICLR)*

[3] Arjovsky, M., Chintala, S., & Bottou, L. (2017, July). Wasserstein generative adversarial networks. In International conference on machine learning (pp. 214-223). PMLR.

[4] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein gans. Advances in neural information processing systems, 30.

[5] Computing The Norm of a matrix, `https://kconrad.math.uconn.edu/blurbs/linmultialg/matrixnorm.pdf`, University of Connecticut.

[6] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[7] Rudin, Walter. *Real and Complex Analysis.* : McGraw-Hill Science/Engineering/Math, 1986.

[8] Herbert Federer. *Geometric measure theory.* Classics in Mathematics. Springer-Verlag Berlin Heidelberg, 1969.

[9] Villani, C. (2009). *Optimal transport: old and new* (Vol. 338, p. 23). Berlin: springer.

[10] Piotr Hajłasz, *Geometric Analysis*, `https://sites.pitt.edu/~hajlasz/Teaching/Math2304Spring2014/Part\%201.pdf`, University of Pittsburgh.

[11] James T. Murphy, *An elementary proof of Rademacher's theorem*, `https://intfxdx.com/downloads/rademacher-thm-2015.pdf` THEOREM

[12] Jordan, M., & Dimakis, A. G. (2020). Exactly computing the local lipschitz constant of relu networks. *Advances in Neural Information Processing Systems*, 33, 7344-7353.

[13] Bhowmick, A., D'Souza, M., & Raghavan, G. S. (2021, September). Lipbab: computing exact lipschitz constant of relu networks. *In International Conference on Artificial Neural Networks* (pp. 151-162). Springer, Cham.

[14] Beer, G., & Hoffman, M. J. (2013). The Lipschitz metric for real-valued continuous functions. *Journal of Mathematical Analysis and Applications*, 406(1), 229-236.

[15] Carlini, N., & Wagner, D. (2017, May). Towards evaluating the robustness of neural networks. In 2017 *ieee symposium on security and privacy (sp)* (pp. 39-57). Ieee.

[16] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.