Theoretical Machine Learning: Generalization and Convex Optimization

Ruixin Guo

Department of Computer Science
Kent State University

Aug 17, 2023

Contents

Contents

The Goal of Machine Learning

Formal Definition of Dataset:

Let $S = \{(x_i, y_i)\}_{i=1}^{n}$ be a dataset of n samples. $x_i \in \mathbb{R}^d$ is the feature vector, and $y_i \in \mathbb{R}$ is the label.

All the samples (x_i, y_i) are drawn iid from an unknown distribution $\mathcal{D}$.

There is an unknown function $h : \mathbb{R}^d \to \mathbb{R}$ such that for every x_i, $h(x_i) = y_i$.

The General Idea of Machine Learning:

We want to find a function f to approximate h such that we can use f instead of h to make prediction for the samples from $\mathcal{D}$.

However, the difficulty in making such approximation is we do not know h and $\mathcal{D}$. What we only have is a dataset S. Machine learning finds f using S in the following way:
(1) Choose a hypothesis space $\mathcal{F}$.
(2) Search f in $\mathcal{F}$ that can make highly accurate prediction on the dataset S.
(3) We expect f can also make accurate prediction on any (x_i, y_i) sampled from $\mathcal{D}$. In other words, f can be generalized on $\mathcal{D}$.

Hypothesis Space

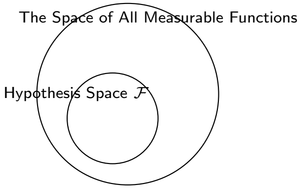
Hypothesis Space: Usually we search for f in a hypothesis space $\mathcal{F}$, and each function f in the $\mathcal{F}$ is called a hypothesis.

The hypothesis space is a subspace of all measurable functions. "hypothesis" means the "shape" of the function pre-determined.

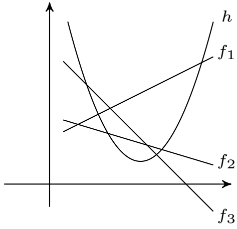
For example, we can let $\mathcal{F} = \{f(x_1, x_2; \theta_1, \theta_2, \theta_3) : \theta_1 x_1 + \theta_2 x_2 = \theta_3\}$ be the space of all linear functions. $\theta_1, \theta_2, \theta_3$ are parameters. By choosing a specific combination of $\theta_1, \theta_2, \theta_3$, we can determine a hypothesis.

However, because we assume f is linear, if h is an quadratic function, then there is no way to find a f in $\mathcal{F}$ to approximate h well. This means different hypothesis space have different ability to learn a target function.

Typically, if a hypothesis space contains more parameters, then the hypothesis will have more complex shape and will be able to learn a complex target function.


The Space of All Measurable Functions
Hypothesis Space $\mathcal{F}$

Hypothesis Space is a subspace of the space of all measurable functions.



Linear functions can never approximate a quadiatic function well.

Loss, Risk and Empirical Risk

Loss Function: The loss function is the error of the prediction of f on a single sample. Using binary loss for example, we define

$$L(x_i, y_i) = \mathbf{1}_{[f(x_i) \neq y_i]}$$

where $\mathbf{1}_{[\text{statement}]} = 1$ if statement is true and $\mathbf{1}_{[\text{statement}]} = 0$ if statement is flase.

Risk Function: The Risk function (Let's call it True Risk), is the average loss for the entire distribution $\mathcal{D}$.

$$R^{\text{true}}(f) = \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}}[L(x_i, y_i)] = \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}}[\mathbf{1}_{[f(x_i) \neq y_i]}]$$

However, because $\mathcal{D}$ is unknown, we cannot calculate True Risk. So we calculate Empirical Risk instead.

Empirical Risk Function: The Empirical Risk Function is the average loss for the finite samples from $\mathcal{D}$. For the samples in the dataset $S = \{(x_i, y_i)\}_{i=1}^{n}$, the Empirical Risk Function is defined as

$$R^{\text{emp}}(f) = \frac{1}{n} \sum_{i=1}^{n} L(x_i, y_i) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{[f(x_i) \neq y_i]}$$

Approximation Error and Estimation Error

Most problems in Machine Learning can be formalized as empirical risk minimization (Let's not consider regularizer first):

$$f_m = \operatorname*{argmin}_{f \in \mathcal{F}} R^{\text{emp}}(f)$$

Suppose the best function in $\mathcal{F}$ is f^*, which minimizes the true risk:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} R^{\text{true}}(f)$$

Let $f^\star = \arg\min_f R^{\text{true}}(f)$ and $R^* = R^{\text{true}}(f^\star)$, where f is from all measurable functions. R^* is the theoretical infimum risk, also known as Bayes Risk[1].

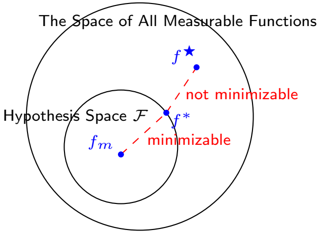
We would like to know how far $R^{\text{true}}(f_m)$ is from R^*:

$$R^{\text{true}}(f_m) - R^* = \underbrace{[R^{\text{true}}(f^*) - R^*]}_{\text{Approximation Error}} + \underbrace{[R^{\text{true}}(f_m) - R^{\text{true}}(f^*)]}_{\text{Estimation Error}}$$

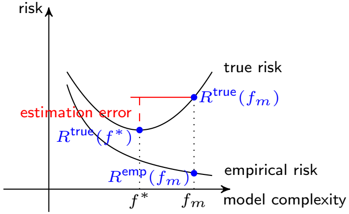
[1] The inherent noise in the data, caused by the ambiguity of data distribution, cannot be minimized.

$$R^{\text{true}}(f_m) - R^* = \underbrace{[R^{\text{true}}(f^*) - R^*]}_{\text{Approximation Error}} + \underbrace{[R^{\text{true}}(f_m) - R^{\text{true}}(f^*)]}_{\text{Estimation Error}}$$

Approximation Error is caused by the unknown of $f^\star$. We can never choose a correct hypothesis space $\mathcal{F}$ because we never know if $f^\star$ is in it, so the approximation error is unavoidable and not minimizable. Estimation Error is caused by the unknown of $\mathcal{D}$. We calculate f using limited samples from $\mathcal{D}$. If we add more samples, estimation error can be minimized.



The Space of All Measurable Functions
$f^\star$
not minimizable
Hypothesis Space $\mathcal{F}$
f^*
f_m
minimizable



Lemma: The estimation error can be bounded by

$$R^{\text{true}}(f_m) - R^{\text{true}}(f^*) \le 2 \sup_{f\in\mathcal{F}} |R^{\text{true}}(f) - R^{\text{emp}}(f)|$$

We call $|R^{\text{true}}(f) - R^{\text{emp}}(f)|$ the generalization gap. Lemma tells us, if we can bound the generalization gap, then we can bound the estimation error.

Generalization Bound

How do we bound $|R^{\text{true}}(f) - R^{\text{emp}}(f)|$? Note that $R^{\text{true}}(f) = \mathbb{E}[R^{\text{emp}}(f)]$. We can consider $R^{\text{emp}}(f)$ that deviates from its mean. Thus, we can use concentration inequality to give a probability bound for $|R^{\text{true}}(f) - R^{\text{emp}}(f)|$.

The concentration inequality includes Markov inequality, Chebyshev inequality, Hoeffding inequality. We choose Hoeffding inequality since it gives a tighter bound.

Hoeffding Inequality: Let $X_1, ..., X_n$ be n iid random variables sampled from a distribution $\mathcal{X}$. Each $X_i \in [a, b]$ where a, b are constants and $a \leq b$. Let $\mathbb{E}[X]$ be the expectation of X_i. Then for all $\epsilon > 0$ we have:

$$P\left[\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mathbb{E}[X]\right| \geq \epsilon\right] \leq 2\exp\left(-\frac{2n\epsilon^2}{(b-a)^2}\right)$$

Since $R^{\text{emp}}(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{[f(x_i)\neq y_i]}$, $R^{\text{true}}(f) = \mathbb{E}_{S\sim\mathcal{D}}[R^{\text{emp}}(f)] = \mathbb{E}_{S\sim\mathcal{D}}[\mathbf{1}_{[f(x_i)\neq y_i]}]$. Let $X_i = \mathbf{1}_{[f(x_i)\neq y_i]} \in [a, b]$ where $a = 0, b = 1$, and let $\delta = 2\exp\left(-\frac{2n\epsilon^2}{(b-a)^2}\right)$, then we get the Hoeffding Bound for $|R^{\text{true}}(f) - R^{\text{emp}}(f)|$:

$$P\left[\left|R^{\text{true}}(f) - R^{\text{emp}}(f)\right| \leq \sqrt{\frac{\log\frac{2}{\delta}}{2m}}\right] \geq 1 - \delta$$

Uniform Convergence

This bound

$$P\left[\left|R^{\text{true}}(f) - R^{\text{emp}}(f)\right| \leq \sqrt{\frac{\log\frac{2}{\delta}}{2m}}\right] \geq 1-\delta$$

is the Hoeffding Bound for a single function f.

Since $R^{\text{true}}(f)$ depends on f, $R^{\text{emp}}(f)$ depends on f and S. Let $L(f,S) = |R^{\text{true}}(f)$ $-R^{\text{emp}}(f)|$, suppose the probability density function of L is $D_{(f,S)}(L(f,S))$. The above bound is for the marginal distribution $D_S(L(f,S))$, because it is conditioned on f. However, our goal is to bound $D_{(f,S)}(L(f,S))$. Since $L(f,S)$ depends on f, we may not have $D_S(L(f,S)) = D_{(f,S)}(L(f,S))$. Hence, the above bound is not useful.

The solution is to bound the quantity $G(S) = \sup_{f\in\mathcal{F}}\{|R^{\text{true}}(f) - R^{\text{emp}}(f)|\}$. The quantity $\sup_{f\in\mathcal{F}}\{|R^{\text{true}}(f) - R^{\text{emp}}(f)|\}$ is independent of f because we take the supremum for $f\in\mathcal{F}$. Now we want to find a bound for the distribution $D_S(G(S))$.

Note that $\sup_{f\in\mathcal{F}}\{|R^{\text{true}}(f) - R^{\text{emp}}(f)|\} \geq C \Leftrightarrow \{\exists f\in\mathcal{F} : |R^{\text{true}}(f) - R^{\text{emp}}(f)| \geq C\} \Leftrightarrow \cup_{f\in\mathcal{F}}(|R^{\text{true}}(f) - R^{\text{emp}}(f)| \geq C)$. So $G(f,S)$ is an union bound.

Uniform Convergence

Theorem: Let $|\mathcal{F}| = N$ be finite, we have the union bound

$$P\left[\sup_{f\in\mathcal{F}}\{|R^{\text{true}}(f) - R^{\text{emp}}(f)|\} \leq \sqrt{\frac{\log N + \log\frac{2}{\delta}}{2n}}\,\right] \geq 1-\delta$$

Uniform Convergence: The uniform convergence means, for any $f \in \mathcal{F}$, $n \to \infty$ will make $R^{\text{true}}(f) - R^{\text{emp}}(f)$ converges to 0 in probability. That is

$$\lim_{n\to} P\left[\sup_{f\in\mathcal{F}}\{|R^{\text{true}}(f) - R^{\text{emp}}(f)|\} \geq \epsilon\right] = 0$$

VC Dimension

When the cardinality of $\mathcal{F}$ is infinite, we use finite measures to describe the complexity of $\mathcal{F}$. Such finite measures include VC Dimension, Rademacher Complexity, Covering Number, etc. I will introduce VC Dimension only.

The key idea of VC Dimension is that we can partition $\mathcal{F}$ into groups. Given a finite number of samples. We can use the number of groups as a measure of the complexity of $\mathcal{F}$.

Given n samples $x_1, x_2, x_3, ..., x_n$, and let $f(x) \in \{-1, 1\}$ be a binary classification function. Let $\{f(x_1), f(x_2), ..., f(x_n)\} = \{-1, 1\}^n$ be the predictions generated by f on the dataset.

Definition (Growth Function): The maximum number of ways that the functions in $\mathcal{F}$ can classify n samples is the Growth Function of $\mathcal{F}$, denoted as $S_{\mathcal{F}}(n)$.

Here if two functions in $\mathcal{F}$ classify the n samples in the same way, we put them into one group. Hence $\mathcal{F}$ can be partitioned into at most 2^n groups. We can use the number of groups as a measure for the complexity of f. Since n is finite, the number of groups is always finite.

VC Dimension

Using Growth Function as a complexity measure of $\mathcal{F}$, we have the following generalization bound.

Theorem (Vapnik-Chervonenkis): For any $\delta > 0$, with respect to a random draw of the data,

$$P\left[\sup_{f\in\mathcal{F}}\left\{R^{\text{true}}(f) - R^{\text{emp}}(f)\right\} \leq 2\sqrt{2\frac{\log S_{\mathcal{F}}(2n) + \log\frac{4}{\delta}}{n}}\right] \geq 1-\delta$$

The $S_{\mathcal{F}}(2n)$ in the above inequality is usually hard to compute, we can upper bound it by VC Dimension.

Definition (Shattering): We say $\mathcal{F}$ shatters an m-point dataset if $S_{\mathcal{F}}(m) = 2^m$.

Definition (VC Dimension): The VC Dimension of a function class $\mathcal{F}$ is the largest s such that $S_{\mathcal{F}}(s) = 2^l$.

For example, suppose we have 3 points x_1, x_2, x_3. For any functions f in $\mathcal{F}$, the classification $\{f(x_1), f(x_2), f(x_3)\}$ has 4 cases :$\{-1, 1, -1\}$, $\{-1, 1, 1\}$, $\{1, -1, -1\}$, $\{-1, -1, 1\}$, then we say the growth function of $\mathcal{F}$ is 4. However, note that x_2, x_3 is shattered by $\mathcal{F}$, so the VC Dimension of $\mathcal{F}$ is 2.

VC Dimension

Theorem (Sauer's Lemma): Let $\mathcal{F}$ be a function class of binary output functions and its VC dimension is s. Then for all $n \in \mathbb{N}$:

$$S_{\mathcal{F}}(n) \leq \sum_{i=0}^{h} \binom{n}{i}$$

Furthermore, for all $n \geq s$, we have

$$S_{\mathcal{F}}(n) \leq (\frac{en}{s})^s$$

By combining Vapnik-Chervonenkis Theorem and Sauer's Lemma, we have

$$P\left[\sup_{f \in \mathcal{F}} \left\{R^{\text{true}}(f) - R^{\text{emp}}(f)\right\} \leq 2\sqrt{2\frac{s \log \frac{2en}{s} + \log \frac{4}{\delta}}{n}}\,\right] \geq 1 - \delta$$

Problems in Uniform Convergence Bound

- In practice, deep neural network generalizes well, although they are overparameterized.
- Traditional uniform convergence suggests that overparameterized model should have greater generalization gap, hence it cannot explain why overparameterized deep neural network generalizes so well. The reason is that uniform convergence bound is vacuous.
- In fact, any bound based on uniform convergence is proved to be unable to capture the generalization gap of overparameterized neural network on simple examples [2].
- Tighter generalization bound needs to be proposed to explain this phenomenon.

[2] Vaishnavh Nagarajan and J Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. Advances in Neural Information Processing Systems, 32, 2019.

Recent Works

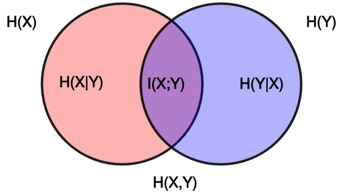
Daniel Russo and James Zou. Controlling bias in adaptive data analysis using information theory. Artificial Intelligence and Statistics, pages 1232–1240. PMLR, 2016.

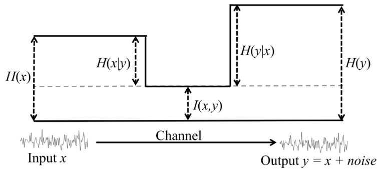
The uniform convergence bound is vacuous because it is for all functions in $\mathcal{F}$. The bound only depends on the hypothesis space $\mathcal{F}$ and it is data-independent and algorithm independent.

However, the machine learning will find only one function f in $\mathcal{F}$. We only need to consider the generalization of one function instead of all functions in $\mathcal{F}$! Since the function is found with given data, Russo and Zou proposed that we can use the information of data to give a generalization bound for f.

Definition (Mutual Information): Let (X, Y) be a pair of random variables over the space $\mathcal{X} \times \mathcal{Y}$, their joint distribution is $P_{(X,Y)}$ and marginal distribution is P_X and P_Y, the mutual information is defined as

$$I(X;Y) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} P_{(X,Y)}(x,y) \log \left(\frac{P_{(X,Y)}(x,y)}{P_X(x) P_Y(y)} \right)$$





Intuitively, mutual information measures the information that X and Y share: It measures how much knowing one of these variables reduces uncertainty about the other.

We can consider a machine learning model f as a channel between the input and output. The mutual information of the input and output can be a metric for the learning ability of f. For example, if f has no ability to learn the data, it will give a random output, thus the input and output will be independent, their mutual information is 0, which means the learning ability of f is 0. The larger channel $I(X;Y)$ means the function f has ability to learn more features, which is analogous to a larger $\mathcal{F}$.

Theorem (Russo and Zou): Let X be the input, n be the size of X, f be the machine learning model, $I(f(X);X)$ be the input-output mutual information. Suppose $R^{\text{emp}}(f) \in [0,1]$ then the generalization bound is

$$\mathbb{E}|R^{\text{true}}(f) - R^{\text{emp}}(f)| \leq \sqrt{\frac{2}{n} I(f(X);X)}$$

Thomas Steinke and Lydia Zakynthinou. Reasoning about generalization via conditional mutual information. Conference on Learning Theory, pages 3437–3452. PMLR, 2020.

Steinke and Zakynthinou pointed that the Input-Out Mutual Information (IOMI) can be infinity if the dataset is in some worst case distribution. Instead, they proposed using Conditional Mutual Information (CMI) instead of IOMI to give a generalization bound, where CMI is finite if n is finite.

CMI considers the input data as $2n$ independent draws from the distribution – namely the n input data points mixed with n "ghost" data points – and measuring how well it is possible to distinguish the true inputs from their ghosts.

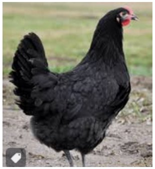
Definition (Conditional Mutual Information, CMI): Let $\tilde{X}$ be input set of $2n$ samples (n true samples + n ghost samples) drawn from a distribution $\mathcal{D}$, and $S = \{0, 1\}^n$ be a set of indexes independent from f and $\tilde{X}$. $S_i = 0$ means the i-th sample is the true one and $S_i = 1$ means the i-th sample is the ghost one. $\tilde{X}_S$ is a subset of $\tilde{X}$ of n samples based on the choice. Then CMI is defined as

$$\mathsf{CMI}_{\mathcal{D}}(f) = I(f(\tilde{X}_S); S|\tilde{X})$$

CMI is bounded by $0 \leq \mathsf{CMI}_{\mathcal{D}}(f) \leq n \log 2$.

To explain what CMI is, consider the following example, we would like to distinguish a true sample image from its ghost sample. Suppose we have two images, one is true sample and one is ghost sample.

True Sample


Ghost Sample

Function f_1: The two images are in the same class, because they are both chicken.

Function f_2: The two images are in different classes, because one is yellow chicken and another is black chicken.

We can say CMI$(f_2) >$ CMI(f_1), because f_2 recognizes both color and animal information while f_1 only recognizes animal information. In other words, as a channel, f_2 shares more information between input and output than f_1. Hence f_2 has stronger learning ability than f_1.

Theorem (Steinke and Zakynthinou): Using CMI, suppose $R^{\text{emp}}(f) \in [0,1]$, the generalization bound is

$$\mathbb{E}_{X\sim\mathcal{D}^n,f}|R^{\text{true}}(f) - R^{\text{emp}}(f)| \leq \sqrt{\frac{2}{n}\text{CMI}_{\mathcal{D}}(f)}$$

Literature Review

1 The VC Dimension, Rademacher Complexity and Covering Number of deep neural network

Paper	Contribution
Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. The Journal of Machine Learning Research, 20(1):2285–2301, 2019.	Proved tighter upper and lower bounds of VC-Dimension of deep neural network with ReLU activation. Let W be the number of parameters and L be the number of layers, the upper bound is $O(WL\log W)$ and the lower bound is $\Omega(WL\log(W/L))$.
Lan V Truong. Generalization error bounds on deep learning with Markov datasets. Advances in Neural Information Processing Systems, 35:23452–23462, 2022.	Proposed a generalization bound based on Rademacher complexity for deep neural network with Markov datasets.
Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. International Conference on Machine Learning, pages 254–263. PMLR, 2018.	Proposed a tigher generalization bound based on compression approach, which is similar to covering number but without an assumption of hypothesis space.

Literature Review

2 New complexity measures and generalization bounds for deep neural network

2.1 Measures based on Information Theory

Paper	Contribution
Amir Asadi, Emmanuel Abbe, and Sergio Verdu. Chaining mutual information and tightening generalization bounds. Advances in Neural Information Processing Systems, 31, 2018.	Proposed a tighter generalization bound for over-parameterized deep neural network by combining chaining and mutual information.
Thomas Steinke and Lydia Zakynthinou. Reasoning about generalization via conditional mutual information. Conference on Learning Theory, pages 3437–3452. PMLR, 2020.	Pointed out that mutual information is unbounded and can be infinity. Addressed this shortcoming by introducing the conditional mutual information for generalization bound.
Mahdi Haghifam, Jeffrey Negrea, Ashish Khisti, et al. Sharpened generalization bounds based on conditional mutual information and an application to noisy, iterative algorithms. Advances in Neural Information Processing Systems, 33:9925–9935, 2020.	Proved that the bound based on conditional mutual information is always tighter than input-output mutual information. And proposed a tighter bound based on disintegrated mutual information.

Literature Review

Paper	Contribution
Gergely Neu, Gintare Karolina Dziugaite, Mahdi Haghifam, and Daniel M Roy. Information-theoretic generalization bounds for stochastic gradient descent. Conference on Learning Theory, pages 3526–3545. PMLR, 2021.	Proposed a generalization bound based on information theory to describe the generalization error with the number of SGD iterations.
Hrayr Harutyunyan, Maxim Raginsky, Greg Ver Steeg, and Aram Galstyan. Information-theoretic generalization bounds for black-box learning algorithms. Advances in Neural Information Processing Systems, 34:24670–24682, 2021.	Proposed that for a trained model, using the predictions of the model on test dataset instead of the output on training set can result in tighter bounds over the existing information-theoretic bounds.
Kenji Kawaguchi, Zhun Deng, Xu Ji, and Jiaoyang Huang. How does information bottleneck help deep learning? International Conference on Machine Learning, 2023.	Proposed the first rigorous generalization bound for information bottleneck. This bound is also based on input-output mutual information but tighter than prior bounds.

Literature Review

2.2 Measures based on Geometry

Paper	Contribution
Umut Simsekli, Ozan Sener, George Deligiannidis, and Murat A Erdogdu. Hausdorff dimension, heavy tails, and generalization in neural networks. Advances in Neural Information Processing Systems, 33:5138–5151, 2020.	Proposed that the Hausdorff dimension of the trajectory of an optimization algorithm can be a complexity measure for the neural network. And proposed a generalization bound based on Hausdorff dimension that does not grow with the number of parameters of neural network.
Tolga Birdal, Aaron Lou, Leonidas J Guibas, and Umut Simsekli. Intrinsic dimension, persistent homology and generalization in neural networks. Advances in Neural Information Processing Systems, 34:6776–6789, 2021.	Proposed that Persistent Homology Dimension of the trajectory of an optimization algorithm can be a complexity measure for a neural network. Comparing with prior works, this approach does not require any additional geometrical or statistical assumption on training dynamics.

Literature Review

Paper	Contribution
Benoit Dherin, Michael Munn, Mihaela Rosca, and David Barrett. Why neural networks find simple solutions: The many regularizers of geometric complexity. Advances in Neural Information Processing Systems, 35:2333–2349, 2022.	Defined a new generalization measure for deep neural network: Geometric Complexity, and showed that various regularization methods all act to reduce geometric complexity, which explains why regularization make better generalization.
Benjamin Dupuis, George Deligiannidis, and Umut Simsekli. Generalization bounds using data-dependent fractal dimensions. International Conference on Machine Learning, 2023.	Proposed a generalization bound without the assumption of Lipschitz continuity. Instead, it uses the fractal dimension of a data-dependent hypothesis space to define the bound.

Contents

Convex Optimization

Convex Problem: A convex problem is an optimization problem of the form

$$\min_{x \in C} f(x)$$

where f is a convex function and C is a convex set.

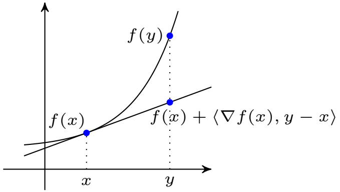
Convex Set: If for any $x, y \in C$, $\lambda x + (1-\lambda)y \in C$ for any $\lambda \in [0,1]$, then C is a convex set.

Convex Function: Let $C \subseteq \mathbb{R}^d$ be a convex set, a function $f : C \to \mathbb{R}$ is convex if for all $x, y \in C$ and $0 \leq \lambda \leq 1$

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$$

Lemma: If f is a convex function, then for any $x, y \in C$,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle$$



Intuition: Make the tangent of f through the point $(x, f(x))$. For any y, the image of y on f is $f(y)$, and the image of y on the tangent is $f(x) + \langle \nabla f(x), y - x \rangle$.

Assumptions in Convex Optimization

Definition (Lipschitz Smooth): A function $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth if for all $x, y \in \mathbb{R}^d$

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|$$

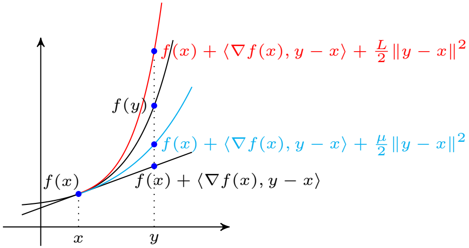
Lemma: If f is convex and L-smooth, then

$$f(y) \le f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|y - x\|^2$$

Definition (Strong Convexity): A function $f : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex if there exists a convex function $g : C \to \mathbb{R}$ such that $f(x) = g(x) + \frac{\mu}{2}\|x\|^2$.

Lemma: If f is μ-strongly convex, then

$$f(y) \ge f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2$$


$f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|y - x\|^2$
$f(y)$
$f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2$
$f(x)$
$f(x) + \langle \nabla f(x), y - x\rangle$
x
y

Intuition of Lemma 2.2 and Lemma 2.3:
Lemma 2.2 says if f is convex and L-smooth, then $f(y)$ will have a convex upper bound $f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|y - x\|^2$.
Lemma 2.3 says if f is μ-strongly convex, then $f(y)$ will have a convex lower bound $f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2$.
Especially, when f is convex but not strongly convex, $\mu = 0$, and the lower bound becomes the tangent.

The steps of Convergence Analysis

Assumption: Make assumptions on f. Usually we assume (1) f is convex and L-smooth or (2) f is μ-strongly convex and L-smooth.

Convergence Proof: Based on the assumption, show that a sequence $\{x_k\}_{k\in\mathbb{N}}$ converges to x^* where $f(x^*) = \inf f$. We need to prove:

There exists a function $g(k)$ such that $f(x_k) - \inf f \leq g(k)$ for any k, and $g(k) \to 0$ when $k \to \infty$.

Evaluation: We use iteration complexity to evaluate optimization algorithms. Suppose $\{x_k\}$ is generated by an optimization algorithm, and let

$$k(\epsilon) = \min\{k : f(x_k) - \inf f < \epsilon\}$$

be the minimum number of iterations to make the error $f(x_k) - \inf f$ be within ϵ. $k(\epsilon)$ is a function of ϵ. When ϵ is fixed, smaller $k(\epsilon)$ means the optimization algorithm converges faster.

The steps of Convergence Analysis

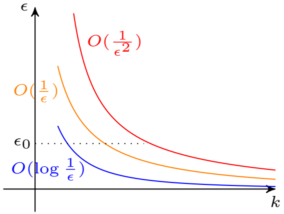
For simplicity, we usually use the Big O notation to describe the order of k. For example, if

$$k(\epsilon) = \frac{c_1}{\epsilon^2} + \frac{c_2}{\epsilon}$$

where c_1 and c_2 are positive constants, then

$$\epsilon(\epsilon) = O(\frac{1}{\epsilon^2})$$

when $0 < \epsilon \ll 1$.



We can compare the convergence speed of optimization algorithms by comparing the order of k. For example. $O(\log\frac{1}{\epsilon})$ is faster than $O(\frac{1}{\epsilon})$, and $O(\frac{1}{\epsilon})$ is faster than $O(\frac{1}{\epsilon^2})$.

Gradient Descent

Gradient Descent based optimization algorithms are most popular in machine learning due to their efficiency.

Algorithm (Gradient Descent, GD): Let $C \subseteq \mathbb{R}^d$ be a convex set, and let the $f : C \to \mathbb{R}$ be a convex function. The Gradient Descent algorithm is as follows:

- Choose $x_0 \in \mathbb{R}^d$ and step size $t > 0$.
- For $i = 0, 1, 2, ...$, define
$$x_{i+1} = x_i - t\nabla f(x_i)$$

Theorem: Consider the settings of Algorithm 2.4. Let $x^* = \arg\min_x f(x)$ and $\inf f = f(x^*)$. Choose the step size $t \leq 1/L$. Then

$$f(x_k) - \inf f \leq \frac{\|x_0 - x^*\|^2}{2tk}$$

Corollary: By Theorem 2.5, the iteration complexity of GD is $O(\frac{1}{\epsilon})$.

Stochastic Gradient Descent and Mini-batch Gradient Descent

Assumption (Sum of Functions): assume $f : \mathbb{R}^d \to \mathbb{R}$ can be written as

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

where each $f_i : \mathbb{R}^d \to \mathbb{R}, i \in \{1, ..., n\}$.

Algorithm (Stochastic Gradient Descent, SGD): Let $x_0 \in \mathbb{R}^d$ be the initial point, and let $\{t_k\}$ be a sequence of step sizes where each $t_k > 0$. The Stochastic Gradient Decent (SGD) algorithm is given by the iterates $\{x_k\}_{k \in \mathbb{N}}$ where:

$$i_k \in \{1, 2, ..., n\} \qquad \text{Sampled with probability } \frac{1}{n}$$

$$x_{k+1} = x_k - t_k \nabla f_{i_k}(x_k)$$

Algorithm (Mini-batch Gradient Descent, Mini-batch GD): Let $b \in \{1, 2, ..., n\}$ be the batch size. The Mini-batch Gradient Decent (Mini-batch GD) algorithm is given by the iterates $\{x_k\}_{k \in \mathbb{N}}$ where:

$$B_k \subset \{1, 2, ..., n\} \qquad \text{Sampled uniformly among sets of size } b$$

$$x_{k+1} = x_k - t_k \nabla f_{B_k}(x_k) \qquad \text{where } f_{B_k}(x_k) = \frac{1}{|B_k|} \sum_{i \in B_k} f_i(x_k)$$

Stochastic Gradient Descent in Machine Learning

In machine learning, out goal is to minimize the empirical risk. Let the empirical risk function be mean square error, we have

$$\min_{\boldsymbol{\theta}\in\Theta} R^{\text{emp}}(f(\boldsymbol{X},\boldsymbol{y};\boldsymbol{\theta})) = \min_{\boldsymbol{\theta}\in\Theta} \frac{1}{n}\sum_{i=1}^{n}(f(x_i;\boldsymbol{\theta}) - y_i)^2$$

where $S = \{(x_i, y_i)\}_{i=1}^{n}$ is the dataset, $\boldsymbol{X} = \{x_1, x_2, ..., x_n\}$, $\boldsymbol{y} = \{y_1, y_2, .., y_n\}$ and $\boldsymbol{\theta} = \{\theta_1, \theta_2, ..., \theta_d\}$ are the parameters of f.

Note that f is determined by $\boldsymbol{\theta}$. So we find the f that minimized R^{emp} by searching $\boldsymbol{\theta}$ using optimization algorithms.

If using GD, the iteration of $\boldsymbol{\theta}$ is

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^{k} - t_k\nabla_{\boldsymbol{\theta}}R^{\text{emp}}(f(\boldsymbol{X},\boldsymbol{y};\boldsymbol{\theta}^{k})) = \boldsymbol{\theta}^{k} - t_k\frac{1}{n}\sum_{i=1}^{n}\nabla_{\boldsymbol{\theta}}(f(x_i;\boldsymbol{\theta}) - y_i)^2$$

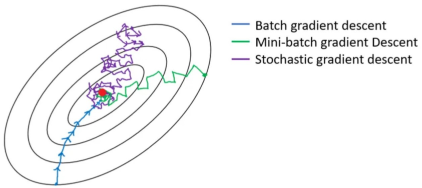
If using SGD, the iteration of $\boldsymbol{\theta}$ is

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^{k} - t_k\nabla_{\boldsymbol{\theta}}(f(x_i;\boldsymbol{\theta}) - y_i)^2$$

It is obvious that the computation cost of calculating gradient in one iteration of GD is n times of SGD.

The Comparison of GD, SGD and Mini-batch GD

Iteration Trajectory: The below figure compares the trajectory of GD, SGD and Mini-batch GD.



Total Cost: Remember that when f is L-smooth and μ-strongly convex, GD converges linearly. We estimate the total cost of the minimum number of steps required to reach an error ϵ.

Algorithm	# Iterations	Cost per iteration[3]	Total cost
GD	$O(\log(\frac{1}{\epsilon}))$	$O(nd)$	$O(nd\log(\frac{1}{\epsilon}))$
SGD	$O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$	$O(d)$	$O(\frac{d}{\epsilon}\log(\frac{1}{\epsilon}))$

This shows when n is large, SGD will have lower total cost than GD. In practice, we use SGD instead of GD when using a large dataset.

[3] The computation of $\nabla f_i(x)$ costs $O(d)$ when $x \in \mathbb{R}^d$, and $\nabla f(x)$ costs $O(nd)$.

Momentum

Momentum and Adaptive Learning Rate are two ways to accelerate the iteration algorithms based on gradient descent.

Algorithm (Heavy Ball Momentum): Let $x^0 \in \mathbb{R}^d$ and $m^{-1} = 0$. Let $\{t_k\}_{k\in\mathbb{N}} \subset (0, +\infty)$ be a sequence of step sizes, and let $\{\beta_k\}_{k\in\mathbb{N}} \subset [0, 1]$ be a sequence of momentum parameters. The Momentum algorithm defines a sequence $\{x^k\}_{k\in\mathbb{N}}$ satisfying for every $k \in \mathbb{N}$,

$$\begin{aligned} m^k &= \beta_k m^{k-1} + \nabla f_{i_k}(x^k) \\ x^{k+1} &= x^k - t_k m^k \end{aligned}$$

Comparing with SGD, the Heavy Ball Momentum uses an momentum term m^k instead of $\nabla f_{i_k}(x^k)$ for gradient descent. Suppose $\beta_k = \beta$ for any k. We can expand m^k as

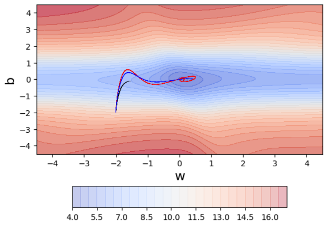
$$m^k = \sum_{j=1}^{k} \beta_{k-j} \nabla f_{i_j}(x^j)$$

So the momentum term m^k is basically an Exponential Moving Average of all past gradients.

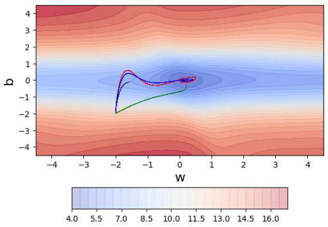
Adaptive Learning Rate

Sparse Gradient Problem: When running stochastic gradient descent with real dataset, we calculate a set of gradients $\nabla f_{i_1}(x_1), \nabla f_{i_2}(x_2), ..., \nabla f_{i_n}(x_n)$ where each $\nabla f_{i_k}(x_k) \in \mathbb{R}^d$. Let $w \in \{1, 2, ..., d\}$, we say that the gradient is sparse on the w-th dimension if the w-th dimensional elements of most gradients is 0.

Suppose the gradients are sparse on dimension w and dense on dimension b, then the trajectory of gradient descent will be as follows



Without Adaptive Learning Rate



With Adaptive Learning Rate (Green)

The convergence on dimension b is faster than dimension w. This is because in most iterations the gradient on w is 0. The convergence of the algorithm depends on the slowest dimension.

Adaptive Learning Rate Algorithms

Adaptive Learning Rate: Assign different learning rate on different dimension. A dense dimension will have smaller learning rate and a sparse dimension will have larger learning rate.

Algorithm (AdaGrad): Let $\{\alpha_t\}_{t\in\mathbb{N}}$ be a sequence of step sizes. Initialize $v_{-1} = 0$. AdaGrad defines a sequence $\{x_t\}_{t\in\mathbb{N}} \in \mathbb{R}^d$ satisfying for every $t \in \mathbb{N}$,

$$v_t = v_{t-1} + (\nabla f_t(x_t))^2$$
$$x_{t+1} = x_t - \alpha_t \frac{\nabla f_t(x_t)}{\sqrt{v_t + \epsilon}}$$

Where the square $(\cdot)^2$, square root $\sqrt{\cdot}$ and division $\div$ of vectors are all element-wise. ϵ is a nonzero vector with trivial values to prevent the calculation failure when some elements of v_t are 0.

So the learning rate of $\nabla f_t(x_t) \in \mathbb{R}^d$ is $\frac{\alpha_t}{\sqrt{v_t+\epsilon}} \in \mathbb{R}^d$. Let $g_{t,i}$ be the element of i-th dimension of $\nabla f_t(x_t)$, then the learning rate of i-th dimension of $\nabla f_t(x_t)$ is $\frac{\alpha_t}{\sqrt{\sum_{j=0}^{t} g_{j,i}^2}}$.
If $g_{j,i} = 0$ for most j, the denominator will be small, thus gives dimension i a large learning rate.

Adaptive Learning Rate Algorithms

Algorithm (RMSProp): Let $\{\alpha_t\}_{t\in\mathbb{N}}$ be a sequence of step sizes. Initialize $v_{-1} = 0$. $\beta \in [0,1]$ be the momentum parameter. RMSProp defines a sequence $\{\alpha_t\}_{t\in\mathbb{N}} \in \mathbb{R}^d$ satisfying for every $t \in \mathbb{N}$,

$$\begin{aligned} v_t &= \beta v_{t-1} + (1-\beta)(\nabla f_t(x_t))^2 \\ x_{t+1} &= x_t - \alpha_t \frac{\nabla f_t(x_t)}{\sqrt{v_t + \epsilon}} \end{aligned}$$

Algorithm (Adam): Let $\{\alpha_t\}_{t\in\mathbb{N}}$ be a sequence of step sizes, and $\beta_1, \beta_2 \in [0,1)$ be the momentum parameters. Initialize $v_0 = 0, m_0 = 0$ and the starting point $x_1 \in \mathbb{R}^d$. Adam defines a sequence $\{x_t\} \in \mathbb{R}^d$ satisfying for every $t \geq 1$,

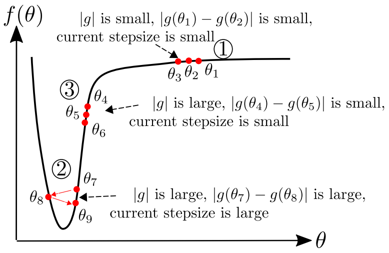
$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla f_t(x_t) \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\nabla f_t(x_t))^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t}\ , \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \\ x_{t+1} &= x_t - \alpha_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \end{aligned}$$

Adam and its variants are the default optimization algorithms for training large neural network model, especially for the large language models.

Recent Works

Adam and its variants are the default optimization algorithms for training large neural network model, like LLaMa 2 [4]. Adam is popular because its fast convergence. However, this does not mean Adam is a perfect algorithm. Recent works focus on fixing the problems in Adam.

Juntang Zhuang, Tommy Tang, Yifan Ding, et al. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. Advances in Neural Information Processing Systems, 33:18795–18806, 2020.


$f(\theta)$
$|g|$ is small, $|g(\theta_1) - g(\theta_2)|$ is small, current stepsize is small
①
θ_3 θ_2 θ_1
③
θ_4
θ_5
θ_6
$|g|$ is large, $|g(\theta_4) - g(\theta_5)|$ is small, current stepsize is small
②
θ_7
θ_8
θ_9
$|g|$ is large, $|g(\theta_7) - g(\theta_8)|$ is large, current stepsize is large
θ

This paper pointed out that Adam can make slow convergence in "large gradient, small curvature" case.
Region ①: small gradient, small curvature, both SGD and Adam have small learning rate.
Region ②: large gradient, large curvature, SGD will have large step size. Adam will have small step size because $\sqrt{v_t}$ is large. Thus Adam will be more stable than SGD.

Region ③: large gradient, small curvature. SGD will have large step size. However, Adam will have small step size because of large $\sqrt{v_t}$. This is not what we want!

[4] Hugo Touvron et al. LLaMa 2: Open Foundation and Fine-Tuned Chat Models. arXiv preprint arXiv:2307.09288 (2023).

Recent Works

To solve this problem, Zhang et al proposed AdaBelief.

```
**Algorithm 1:** Adam Optimizer
**Initialize** $\theta_0$, $m_0 \leftarrow 0$ , $v_0 \leftarrow 0$, $t \leftarrow 0$
**While** $\theta_t$ not converged
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
    **Bias Correction**
        $\widehat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}$, $\widehat{v_t} \leftarrow \frac{v_t}{1-\beta_2^t}$
    **Update**
        $\theta_t \leftarrow \prod_{\mathcal{F},\sqrt{\widehat{v_t}}} \left(\theta_{t-1} - \frac{\alpha \widehat{m_t}}{\sqrt{\widehat{v_t}}+\epsilon}\right)$
```

```
**Algorithm 2:** AdaBelief Optimizer
**Initialize** $\theta_0$, $m_0 \leftarrow 0$ , $s_0 \leftarrow 0$, $t \leftarrow 0$
**While** $\theta_t$ not converged
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
    $s_t \leftarrow \beta_2 s_{t-1} + (1 - \beta_2)(g_t - m_t)^2 + \epsilon$
    **Bias Correction**
        $\widehat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}$, $\widehat{s_t} \leftarrow \frac{s_t}{1-\beta_2^t}$
    **Update**
        $\theta_t \leftarrow \prod_{\mathcal{F},\sqrt{\widehat{s_t}}} \left(\theta_{t-1} - \frac{\alpha \widehat{m_t}}{\sqrt{\widehat{s_t}}+\epsilon}\right)$
```

AdaBelief improves Adam by replacing v_t with s_t. v_t is the EMA of g_t^2. s_t is the EMA of $(g_t - m_t)^2$.

Intuitively, s_t will be small if g_t is close to m_t, and will be large if g_t greatly deviates from m_t. This makes Adam takes larger step size in Region ③ but still keep small step size in Region ②.

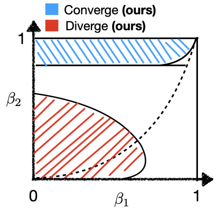
Recent Works

Yushun Zhang, Congliang Chen, Naichen Shi, et al. Adam can converge without any modification on update rules. Advances in Neural Information Processing Systems, 35:28386–28399, 2022.

The iteration rule of Adam includes

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla f_t(x_t)$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla f_t(x_t))^2$$

where $0 < \beta_1 < \sqrt{\beta_2} < 1$. However, Reddi and Kale showed that the wrong choice of β_1 and β_2 can make Adam diverge[5].



Zhang et al proved a bound for β_1, β_2, as shown in the left figure. If β_1 and β_2 are in the red region, Adam will surely diverge; if they are in the blue region, Adam will surely converge.

The default setting of Adam is $(\beta_1, \beta_2) = (0.9, 0.999)$. Zhang el al pointed out that this setting will make Adam converge since it is in the blue region.

[5]Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. International Conference on Learning Representations, 2018.

Literature Review

1 The variants of Adam

The variants of Adam are proposed in pursuit of fixing the bugs of Adam.

Paper	**Contribution**
Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. International Conference on Learning Representations, 2018.	Pointed out that L_2 regularization and weight decay which are equivalent in SGD are not equivalent in Adam, and proposed AdamW [6] which decouples L_2 regularization and weight decay in Adam.
Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. International Conference on Learning Representations, 2018.	Pointed out that the convergence proof of Adam is wrong and Adam does not converge in some cases. Fixed the problem by maintaining a non-decreasing series $\{\hat{v}_t\}$, which gives a variant of Adam – AMSGrad.
Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. International Conference on Learning Representations, 2018.	Pointed out that the generalization of Adam suffer from extreme large and small learning rates, and proposed AdaBound, which bounds the learning rate in each iteration.

[6] This algorithm is used in training a latest large language model LLaMa 2: Hugo Touvron et al. LLaMa 2: Open Foundation and Fine-Tuned Chat Models. arXiv preprint arXiv:2307.09288 (2023).

Literature Review

Paper	Contribution
Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex SGD. Advances in Neural Information Processing Systems, 32, 2019.	Proposed STORM, which uses a variant of momentum term combined with Adam to reduce the variance of gradients. STORM can achieve $O(1/T^{1/3})$ convergence rate for non-convex SGD problem, which is optimal.
Juntang Zhuang, Tommy Tang, Yifan Ding, et al. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. Advances in Neural Information Processing Systems, 33:18795–18806, 2020.	Proposed AdaBelief, which solves the problem that Adam converges slow in "large gradient, small curvature" case.
Byeongho Heo, Sanghyuk Chun, Seong Joon Oh, et al. AdamP: Slowing down the slowdown for momentum optimizers on scale-invariant weights. International Conference on Learning Representations, 2020.	Pointed out that the Batch Normalization in neural network training will decrease the effective step size in momentum, which slows down the convergence. Proposed AdamP which addresses this problem by introducing a projection method in Adam updating rule.

Literature Review

2 Theoretical analysis for Adam

These papers analyze the benefits and shortcomings of Adam.

Paper	Contribution
Liyuan Liu, Haoming Jiang, Pengcheng He, et al. On the variance of the adaptive learning rate and beyond. International Conference on Learning Representations, 2019.	Proved that due to the lack of samples in the early stage, the adaptive learning rate has an undesirably large variance, which makes Adam converge to bad local optima if without warmup.
Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, et al. Why are adaptive methods good for attention models? Advances in Neural Information Processing Systems, 33:15383–15393, 2020.	Empirically showed that Adam outperforms SGD when the norms of the mini-batch gradient is heavy-tailed, which explains why Adam performs better in attention models.
Pan Zhou, Jiashi Feng, Chao Ma, et al. Towards theoretically understanding why SGD generalizes better than Adam in deep learning. Advances in Neural Information Processing Systems, 33:21285–21296, 2020.	Proved that comparing with Adam, SGD is more likely to escape a sharp local minimum and converge to a flat one, which makes it generalize better.

Literature Review

Paper	Contribution
Alexandre Defossez, Leon Bottou, Francis Bach, and Nicolas Usunier. A simple convergence proof of Adam and Adagrad. Transactions on Machine Learning Research, 2022.	Proved that Adam does not converge with default settings. However, the default settings make Adam move away from initial point faster, which may explain its practical success.
Yushun Zhang, Congliang Chen, Naichen Shi, et al. Adam can converge without any modification on update rules. Advances in Neural Information Processing Systems, 35:28386–28399, 2022.	Proved that if $\beta_1 < \sqrt{\beta_2} < 1$ and β_2 is beyond a threshold, Adam will always converge to the neighbor of a critical point. However, improper choice of β_1, β_2 can make Adam diverge.

Contents

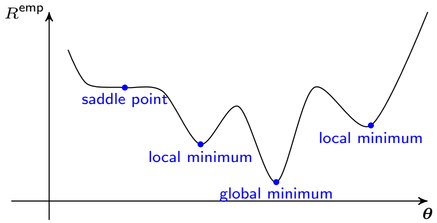
The Goal of Optimization is NOT to Find Global Minimum

In machine learning, optimization algorithms try to find an $f \in \mathcal{F}$ to minimize the empirical risk $R^{\text{emp}}(f)$. We search for $f \in \mathcal{F}$ by searching the parameters $\boldsymbol{\theta} \in \Theta$ using optimization algorithms.

$$\min_{\boldsymbol{\theta} \in \Theta} R^{\text{emp}}(f(\boldsymbol{X}; \boldsymbol{\theta})) = \min_{\boldsymbol{\theta} \in \Theta} \frac{1}{n} \sum_{i=1}^{n} (f(\boldsymbol{x_i}; \boldsymbol{\theta}) - y_i)^2$$

In most cases, the empirical risk $R^{\text{emp}}(f)$ is not convex, because f is not convex. For example, f can be a neural network, and neural networks are known to be non-convex.

For a non-convex $R^{\text{emp}}(f)$, one can only expect to find a saddle point or local minimum. Finding a global minimum is an NP-hard problem. However, it is not necessary to find global minimum. The goal of machine learning is to find the minimizer of $R^{\text{true}}(f)$, which may not be the global minimizer of $R^{\text{emp}}(f)$.



The Curve of Risk and Model Complexity

By Vapnik-Chervonenkis Theorem and Sauer's Lemma, we know

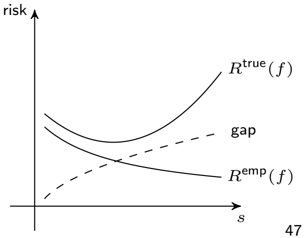
$$P\left[\sup_{f\in\mathcal{F}}\left\{R^{\mathsf{true}}(f)-R^{\mathsf{emp}}(f)\right\}\le 2\sqrt{2\frac{s\log\frac{2en}{s}+\log\frac{4}{\delta}}{n}}\right]\ge 1-\delta$$

where s is the VC Dimension of $\mathcal{F}$, and n is the number of samples. We know two things from the bound:

(1) $R^{\mathsf{emp}}(f)$ is non-increasing with s. This is because, if $\mathcal{F}_1$ is of $s-1$ VC Dimension and $\mathcal{F}_2$ is of s VC Dimension, then $\mathcal{F}_1\subset\mathcal{F}_2$. Let f_1 be the minimizer of $R^{\mathsf{emp}}(f)$ for all functions in $\mathcal{F}_1$, then f_1 must be in $\mathcal{F}_2$. That is, the minimum of $R^{\mathsf{emp}}(f)$ cannot go larger with s increasing.

(2) The upper bound of $\sup_{f\in\mathcal{F}}\left\{R^{\mathsf{true}}(f)-R^{\mathsf{emp}}(f)\right\}$ increases with s. Note that $s\le n$, so $s\log\frac{2en}{s}$ increases with s.

By (1) and (2), the function of $R^{\mathsf{true}}(f)$, $R^{\mathsf{emp}}(f)$ and the gap $\sup_{f\in\mathcal{F}}\left\{R^{\mathsf{true}}(f)-R^{\mathsf{emp}}(f)\right\}$ with respect to s will be like:



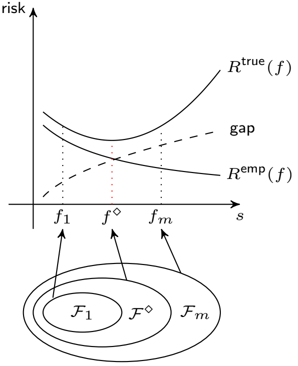
Structural Risk Minimization

Remember that there is a bijection between Θ and $\mathcal{F}$ since every $\boldsymbol{\theta} \in \Theta$ is one-to-one mapped to a $f(\boldsymbol{x}; \boldsymbol{\theta}) \in \mathcal{F}$. We can take a subset of Θ: $\bar{\Theta} = \{\boldsymbol{\theta} \in \Theta \mid \|\boldsymbol{\theta}\| \leq c\}$, where c is a constant. Then $\bar{\Theta}$ corresponds to $\bar{\mathcal{F}}$ and $\bar{\mathcal{F}} \subset \mathcal{F}$. By searching f in $\bar{\mathcal{F}}$ instead of $\mathcal{F}$, we can shrink $\sup_{f \in \mathcal{F}} \{|R^{\text{true}}(f) - R^{\text{emp}}(f)|\}$.

Structural Risk Minimization[7]: Consider $\|\boldsymbol{\theta}\| \leq c$, we let c be a set of values $c_1 < c_2 < \ldots < c_m$. Hence we generated a set of nested hypothesis space

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \ldots \subset \mathcal{F}_m$$

Starting from c_m, suppose that there exists some functions in $\mathcal{F}_m$ that overfits the dataset. By searching from $c_m, c_{m-1}, c_{m-2} \ldots$, we reduce the range of hypothesis space by following $\mathcal{F}_m, \mathcal{F}_{m-1}, \mathcal{F}_{m-2} \ldots$. Eventually we can find a function $f^{\diamond}$ that is close to the minimizer of $R^{\text{true}}(f)$ in a hypothesis space $\mathcal{F}^{\diamond}$.



[7] Vladimir Vapnik. Statistical Learning Theory. Wiley, New York, 1998.

Regularization

Regularization: In practice, we do not bound $\|\boldsymbol{\theta}\|$ explicitly. Instead, we add a regularizer $\lambda\|\boldsymbol{\theta}\|$ to the problem we optimize:

$$\min_{\boldsymbol{\theta}\in\Theta} R^{\text{emp}}(f(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta})) = \min_{\boldsymbol{\theta}\in\Theta} \frac{1}{n}\sum_{i=1}^{n}(f(x_i; \boldsymbol{\theta}) - y_i)^2 + \lambda\|\boldsymbol{\theta}\|$$

where $\|\cdot\|$ is often defined as L_1 norm or L_2 norm square. $\lambda > 0$ is an adjustable parameter, where larger λ means smaller c.

The goal of machine learning is to find f^*, the minimizer of $R^{\text{true}}(f)$. Since R^{true} is unknown to us due to the unknown distribution $\mathcal{D}$, we solve this problem in practice by the following steps:
(1) Split the dataset into training set and test set. Use the training set to train the machine learning model.
(2) Training: Define a over-parameterized hypothesis space $\mathcal{F}$ such that if we simply optimize $R^{\text{emp}}(f)$ with $\lambda = 0$ (without regularization), we are likely to get an overfitting f_l.
(3) Evaluation: Carefully increasing λ and for each fixed λ we train a model. Select the model that gives the smallest error on the test set.