Notes on the Theory of Machine Learning

Yanhua Huang

Oct 2020

This post reviews some basic machine learning theory. In particular, we discuss learnability, complexity, generalization, stability, and convergence.

1 What is Learning?

One of the essential things about machine learning is to understand what is **learning** for the computer. Following Domingos [8], learning consists of three components: representation, evaluation and optimization.

Represetation First of all, any learning algorithm or learned result must be in representation that the computer can handle. Generally, we expect to learn a mapping, e.g., a mapping from the input image to the desired label or a mapping from the data point to the meaningful cluster. Formally, we call the mapping as **concept**. The collection of concepts we want to learn is called **concept class**. Given a learning algorithm, the set of all possible concepts considered is called **hypotheis space**. For example, the decision tree formulates the mapping by a tree structure, and the Naive Bayes classifier assumes strong (naive) independence between the features. Given a problem, we wonder if we can learn it and how complicated is it, corresponding to the learnability (Sec.2) and complexity (Sec.3) in the machine learning theory.

Evaluation Evaluation is necessary for machine learning. On the one hand, we want to achieve generalized performance beyond the training data. Sec.4 will analyze factors, e.g., the hypothesis space's complexity, that may influence the generalization. On the other hand, apart from evaluations like accuracy and recall, it is also necessary to consider the robustness of the learning procedure and the learned concept, especially when facing the perturbation and attack in real-world applications. Sec.5 will demonstrate how to measure the stability of a given learning algorithm.

Optimization Optimization means searching in the hypothesis space to find the optimal concept corresponding to the evaluation method. Most of the optimizations in machine learning is an iterative procedure. There are three fundamental problems for the iteration. Does it converge? Where is the convergence point? What is the rate of convergence? Moreover,

sometimes we optimize a surrogate objective instead of the one produced by the evaluation method, e.g., minimizing the cross-entropy loss instead of maximizing the accuracy in classification tasks. We further wonder about the consistency of the surrogate. Sec.6 tries to answer these questions.

2 Learnability

The well-known learnability theory of machine learning is the **Probably Approximately Correct** (PAC) [23]. PAC provides a framework to describe that the algorithm can learn the correct concept approximately under the perspective of probability. The word "correct" considers the generalization error at the most time. Formally, a target concept class $\mathcal{C}$ is **PAC-learnable** by a hypothesis space $\mathcal{H}$ if for any concept $c \in \mathcal{C}$, any target distribution $\mathcal{D}$, and any $0 < \epsilon, \delta < 1$, dataset D consisting of $m = \mathcal{O}(poly(\frac{1}{\epsilon}, \frac{1}{\delta}, \text{bits}(x), \text{bits}(c)))$ i.d.d. samples from $\mathcal{D}$ is sufficient for some algorithms, s.t. the produced $h \in \mathcal{H}$ satisfies

$$\Pr[\mathcal{L}(h|D) \leq \epsilon] \geq 1 - \delta,$$

where $\mathcal{L}(h|D) = \Pr_{x \sim D}[h(x) \neq c(x)]$.

PAC learnable means with polynomial samples, an arbitrarily small error is achievable with high probability. However, for challenging problems, it is impossible to achieve small enough error. Consequently, we consider the smallest error on the whole hypothesis space, resulting in the **agnostic PAC learnable** [14] by

$$\Pr[\mathcal{L}(h|D) - \min_{h' \in \mathcal{H}} \mathcal{L}(h'|D) \leq \epsilon] \geq 1 - \delta.$$

Note that the complexity of the hypothesis space influences the learnability a lot. It is straightforward to prove that every finite $\mathcal{H}$ is agnostically learnable with $m \in \mathcal{O}(\frac{1}{\epsilon^2} \ln \frac{|\mathcal{H}|}{\delta})$ by the Hoeffding inequality. For infinite hypothesis spaces, it must first know their complexity, as introduced in the next section.

3 Complexity

For finite hypothesis space, it is natural to measure the complexity by its cardinality. This section discusses two complexity measures that can be used for any hypothesis space: the distribution-free method **Vapnik-Chervonenkis (VC) dimension** [24], and the distribution-based method **Rademacher complexity** [2].

The VC-dimension describes the complexity by the ability to shatter the data. Formally, denote the restriction of hypothesis space $\mathcal{H}$ on dataset D is

$$\mathcal{H}_D = \{h(x)|x \in D, h \in \mathcal{H}\}.$$

We say a dataset X is shattered by $\mathcal{H}$ if $|H_X| = 2^{|X|}$. The VC-dimension of a hypothesis space $\mathcal{H}$ is then given by

$$VC(\mathcal{H}) = \max |X|, \text{s.t. } X \text{ is shattered by } \mathcal{H}.$$

Let us consider the VC-dimension of a simple hypothesis space

$$\mathcal{H}_{thr} = \{h(x) = \mathbb{I}(x > t) - \mathbb{I}(x \geq t) | t \in \mathbb{R}\}$$

for one dimensional binary classification problems. It is easy to know its VC-dimension is not less than 1. Consider $D = \{a, b\}$ with $a < b$, any concept in $\mathcal{H}_{thr}$ can't achieve $\{(a, 1), (b, -1)\}$. As a result, the VC-dimension of $\mathcal{H}_{thr}$ is 1.

Another notion of complexity is the Rademacher complexity that considers the data distribution and suitable for any real-valued concept (not only concept mapping to discrete values). The Rademacher complexity is formally given by

$$\mathbb{E}_\sigma[\sup_{h \in \mathcal{H}} \frac{1}{|D|} \sum_{x \in D} \sigma h(x)],$$

where σ is random variable uniformly chosen from $\{-1, 1\}$. Intuitively, the Rademacher complexity describes a maximum achievable correlation of the hypothesis space. In discrete situations, we can view σ as a random label, and the Rademacher complexity is the hypothesis space's ability to handle the desired input signal with random targets.

There are a lot of works about analyzing the complexities of commonly used hypothesis spaces, such as the VC-dimension of SVM [6] and the Rademacher complexity of neural networks [4]. Basically, complex spaces have rich expression but are hard to generalize to the unseen data. We will discuss this topic in the next section.

4 Generalization

As mentioned in the last section, complexity influences the generalization well. Commonly used methods for generalization, such as weight decay and early stopping, are all about controlling the complexity. Given the produced concept h and the target concept c, we analyze their generalization performance by decomposing the difference as follows:

$$\mathcal{L}(h) - \mathcal{L}(c) = \underbrace{\mathcal{L}(h) - \mathcal{L}(\tilde{c})}_{\text{Optimization Error}} + \underbrace{\mathcal{L}(\tilde{c}) - \mathcal{L}(h^*)}_{\text{Estimation Error}} + \underbrace{\mathcal{L}(h^*) - \mathcal{L}(c)}_{\text{Modeling Error}},$$

where $\tilde{c}$ is the optimal concept with respect to the emprical error, h^* is the optimal concept in the hypothesis space with respect to the generalization error.

Basically, the hypothesis space's complexity can trade off between the modeling error and the optimization error. Furthermore, the complexity also can provide an upper bound for the estimation error [2, 24]. However, many works showed that deep nets generalize well despite their high complexity. [1, 19, 26] tried to explain this phenomenon in other aspects.

5 Stability

In this section, we discuss another aspect of the evaluation for the learning algorithm, i.e., the stability to handle perturbations. There two perturbations commonly associated with the stability of machine learning algorithms: sample-level and feature-level perturbation. In particular, for the sample-level, we consider single sample perturbation on the training dataset, and for the feature-level, we focus on adversarial noise.

Sample-level perturbation mainly influent the training procedure. As a result, we can measure the stability as follows:

$$d := \mathbb{E}_i[|\mathcal{L}(h|D) - \mathcal{L}(h|D_i)|],$$

where D_i means the perturbation occurs in the i-th sample of the training dataset D. Generally, this stability is related to the data size, so we say that the learning algorithm is stable if it can achieve $\lim_{|D|\to\infty} d = 0$. Bousquet and Elisseeff [5] proved that commonly used learning methods, such as SVM and least squares regression, hold stability with regularization if the perturbation occurs uniformly. Mukherjee et al. [16] further proved that stability is sufficient for generalization under empirical risk minimization.

Feature-level perturbation occurs during the inference stage. Although deep nets achieve great success in many tasks, their instability limits further applications to security-sensitive fields. Recently, a so-called adversarial attack that refers to imperceptible input changes for humans to test deep nets' stability has aroused wide attention [10]. Formally, we consider the following risk under adversarial noise:

$$\mathbb{E}[\sup_{||\xi||<\epsilon} \mathbb{I}[h(x+\xi) \neq h(x)]],$$

where ϵ is the radius of the noise. Several works viewed this problem under the perspective of generalization and considered the robustness and the accuarcy at the same time [11, 22, 27].

6 Convergence

In machine learning, it is common to solve an optimization problem:

$$\min \mathcal{L}(\theta), \text{s.t. some conditions are satisfied,}$$

where θ is the parameter of the concept, and $\mathcal{L}$ is the loss function. As mentioned before, when the evaluation objective is hard to optimize directly, we instead choose $\mathcal{L}$ to be a surrogate function with good properties for optimization. A natural question is whether learning with the surrogate loss is consistent with learning with the evaluation objective? Formally, if the surrogate loss's result converges to the one lead by the evaluation objective, we say the surrogate loss achieves consistency. Bartlett et al. [3] provided sufficient and necessary conditions for consistency with accuracy. Other works further discussed the situations with AUC [9] and NDCG [20].

We then move to the convergence of the iterative procedure. Let θ_t be the output parameter in t-th iteration of optimization, and θ^* be the optimal parameter in the hypothesis space. We can measure the rate of convergence as follows:

$$\rho := \lim_{t\to\infty} \frac{\mathcal{L}(\theta_{t+1}) - \mathcal{L}(\theta^*)}{\mathcal{L}(\theta_t) - \mathcal{L}(\theta^*)}.$$

If $\rho = 1$, we say it achieves a sublinear rate. If $\rho \in (0, 1)$, we say it achieves a linear rate. If $\rho = 0$, we say it achieves a superlinear rate. Generally, the analysis for the convergence rate usually needs some assumption on the mathematical properties of the loss function, such as convex and smooth. In the rest of this section, we first introduce gradient-based optimization methods and their convergence rate for convex losses, and then turn into the nonconvex situation.

The most basic graident-based optimization method is the gradient descent (GD) algorithm. Formally, GD uses the first-order Taylor expansion to approximate the loss function at the current parameter θ_t:

$$\mathcal{L}(\theta) \approx \mathcal{L}(\theta_t) + \nabla\mathcal{L}(\theta_t)^{\mathrm{T}}(\theta - \theta_t).$$

The minimization then becomes a linear optimization problem with update rule:

$$\theta_{t+1} = \theta_t - \eta\nabla\mathcal{L}(\theta_t),$$

where $\eta > 0$ is also known as the step size. If η is small enough, then $\mathcal{L}(\theta_{t+1}) \leq \mathcal{L}(\theta_t)$.

The main disadvantage of GD is that it can only handle unconstrained and differentiable problems. Projected subgradient methods address this issue by first using subgradient to surrogate the natural gradient and then guaranteeing constraints with projection. In detail, for a convex function f, we say g is the subgradient of f at $x \in \mathrm{dom}f$, if for arbitrary $y \in \mathrm{dom}f$, it satisfies

$$f(y) \geq f(x) + g^{\mathrm{T}}(y - x).$$

For example, arbitrary $g \in [-1, 1]$ is the subgraident of $f(x) = |x|$ at $x = 0$. Note that the subgradient is a generalization of the gradient for convex functions. It is easy to prove that the subgradient is unique and equal to the natural gradient if the natural gradient exists.

The Frank-Wolfe algorithm, also known as the conditional gradient method, improves the efficiency of projected subgradient methods by considering the constraints $\mathcal{W}$ within Taylor expansion:

$$\theta_{t+1} = \arg\min_{\theta \in \mathcal{W}} \nabla \mathcal{L}(\theta_t)^{\mathrm{T}} \theta.$$

Moreover, the Frank-Wolfe algorithm also proposes to apply exponential smoothing when updating the parameter to imporve stability.

Nesterov [18] provided a general acceleration method for smooth function, known as Nesterov acceleration, and further proved that GD with Nesterov acceleration achieves the fastest convergence rate in the first-order smooth functions optimization in theory.

Algorithm	Assumption	Convergence Rate
GD	convex & smooth	sublinear
GD	strong-convex & smooth	linear
Projected subgradient	convex & Lipschitz continuous	sublinear
Projected subgradient	convex & smooth	sublinear
Projected subgradient	strong-convex & smooth	linear
Frank-Wolfe	convex & Lipschitz continuous	sublinear
Frank-Wolfe	convex & smooth	sublinear
Frank-Wolfe	strong-convex & smooth	linear
SGD	convex & Lipschitz continuous	sublinear

Newton's method is a natural expansion of GD with second-order information:

$$\mathcal{L}(\theta) \approx \mathcal{L}(\theta_t) + \nabla \mathcal{L}(\theta_t)^{\mathrm{T}}(\theta - \theta_t) + \frac{1}{2}(\theta - \theta_t)^{\mathrm{T}} \nabla^2 \mathcal{L}(\theta_t)(\theta - \theta_t).$$

If the Hessian matrix $\nabla^2 \mathcal{L}(\theta_t)$ is positive definite, then the approximation above is a convex function with the optimal solution

$$\theta_{t+1} = \theta_t - [\nabla^2 \mathcal{L}(\theta_t)]^{-1} \nabla \mathcal{L}(\theta_t).$$

Newton's method provides a more accurate step size adjustment and a faster convergence rate than first-order methods. However, it introduces more calculations and causes further issues: the Hessian matrix $\nabla^2 \mathcal{L}(\theta_t)$ is expensive and not always positive definite. Quasi-Newton methods address these issues by constructing an iteratively updated postive definite matrix similar to the Hessian matrix as a replacement. Quasi-Newton methods such as BFGS and DFP are able to achieve the same convergence rate as Newton's method with good initialization.

Stochastic gradient descent (SGD) is a stochastic approximation of GD that replaces the actual gradient from the entire dataset by the estimation

from a randomly selected subset. SGD is extremely efficient for large-scale datasets. Similar to SGD, we can obtain a vanilla stochastic Quasi-Newton methods:

$$\theta_{t+1} := \theta_t - \eta \tilde{H} \nabla \tilde{\mathcal{L}}(\theta_t),$$

where η is the learning rate, the Hessian matrix $\tilde{H}$ and the gradient $\nabla \tilde{\mathcal{L}}$ are calculated on the subset of the data. However, this direct method will lead to high variance, which is harmful to the learning procedure. A similar issue also occurs in first-order stochastic optimization methods. There is a lot of research working on variance reduction methods for stochastic optimization, such as adding regularization [7, 12], or improving sampling strategy [17, 28].

So far we disscussed the convergence rate on convex assumption of the objective function. One of the difficulty in nonconvex analysis is to identify the global optimum. For derivate free optimization, it is common to consider the convergence of performance. Zero-order methods, such as cross entropy method and Bayesian optimization, demonstrate their efficiency in many nonconvex problems [21, 25]. For derivate based situations, it is common to consider a weak convergence condition:

$$\min_{t=1,\ldots,T} \mathbb{E}||\nabla \mathcal{L}(\theta)||^2 \to 0,$$

i.e., whether the learning algorithm can reach the zero-gradient stage during training. Note that the first-order stationary point may be a local minimum or a saddle point. Lee et al. [15] proved that gradient descent with random initialization and appropriate step size does not converge to a saddle point. Kawaguchi [13] showed that every local minimum of arbitrary deep linear neural network with the squared loss function is a global minimum with some assumptions on the training data.

References

[1] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

[2] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[3] Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[4] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

[5] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

[6] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[7] Richard H Byrd, Samantha L Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[8] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[9] Wei Gao and Zhi-Hua Zhou. On the consistency of auc pairwise optimization. In *IJCAI*, pages 939–945. Citeseer, 2015.

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[11] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136, 2019.

[12] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[13] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems*, pages 586–594, 2016.

[14] Michael J Kearns, Robert E Schapire, and Linda M Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994.

[15] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on learning theory*, pages 1246–1257, 2016.

[16] Sayan Mukherjee, Partha Niyogi, Tomaso Poggio, and Ryan Rifkin. Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, 25(1-3):161–193, 2006.

[17] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in neural information processing systems*, pages 1017–1025, 2014.

[18] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/k^ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

[19] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of overparametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

[20] Pradeep Ravikumar, Ambuj Tewari, and Eunho Yang. On ndcg consistency of listwise ranking methods. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 618–626, 2011.

[21] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

[22] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[23] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[24] VN Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

[25] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, Nando De Freitas, et al. Bayesian optimization in high dimensions via random embeddings. In *IJCAI*, pages 1778–1784, 2013.

[26] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[27] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019.

[28] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*, pages 1–9, 2015.