Algorithmic Stability and Leave-one-out Cross Validation Bounds

Antong Cheng

May 2023

1 Introduction

The notion of stability is motivated by a simple question. For any mathematical model, given a small change in the input, by how much does the output change? This is a non-trivial question. Many problems, such as the three-body problem, or weather forecasting, are known to have no stable solution. Any small but inevitable error (due to measurement, sampling, etc.) would eventually be magnified over time such that long term prediction is impossible.

When it comes to machine learning, "stability" refers to the model's capability of reproducing its predictions given a small change in the sample. We now provide a few definitions [2] in this idea.

Definition 1. *Given a sample set $S = \{x_1, x_2, ..., x_n\} \subset X$, define $\boldsymbol{S^{|i}}$ to be the modified sample removing the i-th element and $\boldsymbol{S^i}$ to be the modified sample replacing the i-th element. That is,*

$$S^{|i} = \{x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n\},$$

$$S^i = \{x_1, x_2, ..., x_{i-1}, x_i', x_{i+1}, ..., x_n\}.$$

Through the entirety of the article we assume all our algorithms are symmetric. That is, permutations of the sample S generate the same hypothesis h_S.

Definition 2. *An algorithm with loss function L has **hypothesis stability** (β_1, β_2) if $\forall 1 \leq i \leq m$,*

$$\mathbb{P}_S(\|h_S - h_{S^{|i}}\| \geq \beta_2) \leq \beta_1.$$

Where $\|\cdot\|$ denotes any norm on the hypothesis space. For simplicity, we'll assume $\|g(x)\| = \mathbb{P}_{x \in X}(g(x) \neq 0)$ unless stated otherwise.

Definition 3. *An algorithm with loss function L has **error stability** (β_1, β_2) if $\forall 1 \leq i \leq m$,*

$$\mathbb{P}_S(|R(h_S) - R(h_{S^{|i}})| \geq \beta_2) \leq \beta_1.$$

Hypothesis stability is, in fact, a stronger statement than error stability. It was claimed without proof by K and R [2], which we fill in now.

Theorem 1. *If H has hypothesis stability (β_1, β_2), it has error stablity (β_1, β_2).*

Proof. Let f be the target function. Since the algorithm is symmetric and sampling is i.i.d, without loss of generality, let $i = m$.

$$\begin{aligned}
|R(h_S) - R(h_{S^{|m}})| &= |\mathbb{P}_x(h_S \neq f) - \mathbb{P}_x(h_{S^{|m}} \neq f)| \\
&= |\mathbb{E}_x[\mathbb{1}_{h_S \neq f}] - \mathbb{E}_x[\mathbb{1}_{h_{S^{|m}} \neq f}]| \\
&\leq \mathbb{E}_x[|\mathbb{1}_{h_S \neq f} - \mathbb{1}_{h_{S^{|m}} \neq f}|] \\
&= \mathbb{E}_x[\mathbb{1}_{(h_S \neq f \wedge h_{S^{|m}} = f) \vee (h_S = f \wedge h_{S^{|m}} \neq f)}] \\
&= \mathbb{P}_x((h_S \neq f \wedge h_{S^{|m}} = f) \vee (h_S = f \wedge h_{S^{|m}} \neq f)) \\
&\leq \mathbb{P}_x(h_S \neq h_{S^{|m}}) \\
&= \|h_S - h_{S^{|m}}\|
\end{aligned}$$

So

$$\begin{aligned}
\{|R(h_S) - R(h_{S^{|m}})| \geq \beta_2\} &\subset \{\|h_S - h_{S^{|m}}\| \geq \beta_2\} \\
\mathbb{P}_S(|R(h_S) - R(h_{S^{|m}})| \geq \beta_2) &\leq \mathbb{P}_S(\|h_S - h_{S^{|m}}\| \geq \beta_2)
\end{aligned}$$

□

It is notable that we have not yet defined what it means for an algorithm to be "stable" or "unstable", which would be a topic for future sections. At this point, we are simply introducing stability as a tool for analysis. The main way to use it would be to express β_2 as a function of β_1, the latter being a probability measure. We could then pick β_1 to be any arbitrary value in $(0, 1)$ suitable for the proof. This will be our main technique of utilizing this tool.

2 Generalization Error Bound with LOOCV

2.1 Preliminary Results

It is known, due to Vapnik himself, that with arbitrary probability δ, a function with finite VC dimension is error-stable (that is, can obtain arbitrarily small β_2 given arbitrarily small β_1) with $O(\delta, \sqrt{d/m \ln(m/d)})$[1].

Theorem 2. *Let S be a sample of size $m > 4$, H be a hypothesis set with VC dimension $d < m < \infty$, then $\forall 0 < \delta < 1$, $h_S \in H$*

$$\mathbb{P}\left(|R(h_S) - \hat{R}_S(h_S)| \leq 2\sqrt{\frac{d(\ln(m/d) + 1) + \ln(9/\delta)}{m}}\right) \geq 1 - \delta$$

This is to say that empirically, we could approximate the generalization error $R(h_S)$ by the empirical error $\hat{R}_S(h_S)$ with $O\left(\sqrt{d/m\ln(m/d)}\right)$ accuracy. We now present an alternative way to approximate the generalization error. Stability, as it turns out, is a strong tool that helps us measure its effectiveness.

2.2 LOOCV Estimate

Definition 4. *The **Leave-one-out Cross Validation (LOOCV) estimate** is defined as*

$$\hat{R}_{LOO}(h_S) \coloneqq \frac{|\{i \in \{1,...,m\} : h_{S^{|i}}(x_i) \neq f(x_i)\}|}{m}$$

Intuitively, the LOOCV describes the deviation of the hypothesis h_S given a small change in the input sample S. In the lectures, we have seen cases of LOOCV analysis being applied on SVM algorithms, namely the notable fact that it is an unbiased estimator of the generalization error. While this is true, now with the notion of stability as a tool, further analysis can be conducted to derive the bound on the accuracy of this estimate. [3] The rest of the section will be dedicated to proving generalization bounds in terms of LOOCV and comparing its performance against that of empirical error.

Theorem 3. *For any symmetric algorithm with hypothesis stability (β_1, β_2), $\forall\delta \in (0,1)$,*

$$\mathbb{P}\left(|\hat{R}_{LOO}(h_S) - R(h_S)| \leq \sqrt{\frac{1/(2m) + 3(\beta_1+\beta_2)}{\delta}}\right) \geq 1-\delta$$

If we further assume that H is a hypothesis set with VC dimension $d < \infty$ and that the target function is realizable ($f \in H$), then we obtain the following stronger result [2]:

Theorem 4. *Suppose H has VC dimension $d < \infty$, $f \in H$. If we have a symmetric algorithm such that $\hat{R}_S(h_S) = 0$ for all sample S, then $\forall\delta \in (0,1)$*

$$\mathbb{P}\left(|\hat{R}_{LOO}(h_S) - R(h_S)| = O\left(\sqrt{\frac{(d/m)\log(m/d)}{\delta}}\right)\right) \geq 1-\delta$$

This is a general result. We explained earlier at the end of section 1 that we did not define "stable" and "unstable", so this result has no condition dependent on stability. Hypothesis stability is only used as a tool to prove certain intermediate lemmas. As expected, this is a weak result, with an irresolvable $\sqrt{1/\delta}$ multiplier compared with the empirical error bound. If we use error stability as a tool, we could derive another general bound. But before that, we introduce another notion to account for error stability's weaker implication.

Definition 5. *The LOOCV estimate (γ_1, γ_2)-overestimates the empirical error if*

$$\mathbb{P}_S(\hat{R}_{LOO}(h_S) \leq \hat{R}_S(h_S) - \gamma_2) \leq \gamma_1$$

Theorem 5. *Suppose H has VC dimension $d < \infty$ and error stability (β_1, β_2), and LOOCV (γ_1, γ_2)-overestimates the training error for A. Let $\epsilon = 3\beta_1 + \beta_2 + \gamma_1 + \gamma_2$. Then $\forall \delta > 0$,*

$$\mathbb{P}_S\left(|\hat{R}_{LOO}(h_S) - R(h_S)| \leq \left(3\sqrt{\frac{(d+1)(\ln(9m/d)+1)}{m}} + \epsilon\right)/\delta\right) \geq 1 - \delta$$

This is to say that for a good approximation, we require both good error stability (small β_1, β_2) and also that the two empirical estimates, LOOCV and empirical error, mostly agree with each other. This leads to the important following theorem that describes the estimate on an error-minimizing algorithm [2].

Theorem 6. *Suppose H has VC dimension $d < \infty$ and the algorithm minimizes empirical error. Then $\forall \delta \in (0, 1)$,*

$$\mathbb{P}\left(|\hat{R}_{LOO}(h_S) - R(h_S)| \leq \left(8\sqrt{\frac{(d+1)(\ln(9m/d)+2)}{m}}\right)/\delta\right) \geq 1 - \delta$$

Proof. First, we claim that any error-minimizing algorithm's LOOCV $(0, 0)$ overestimates the training error, that is, $\hat{R}_{LOO}(h_S) \geq \hat{R}_S(h_S)$ with $\mathbb{P} = 1$.

Suppose, for the sake of contradiction, that $\exists S \sim D^m, i \in [1, m]$ such that $h_S(x_i) \neq y_i$ but $h_{S^{|i}}(x_i) = y_i$. Now define $\epsilon_S(h) \coloneqq R_S(h) \cdot |S|$. Since the algorithm minimizes empirical error, $\epsilon_{S^{|i}}(h) \geq \epsilon_{S^{|i}}(h_{S^{|i}})$ for all $h \in H$. In particular, it must hold for h_S as well, so

$$\epsilon_{S^{|i}}(h_S) \geq \epsilon_{S^{|i}}(h_{S^{|i}}) \, (= \epsilon_S(h_{S^{|i}}))$$

(Since $h_{S^{|i}}$ is correct on x_i.) Then see that

$$\epsilon_{S^{|i}}(h_S) = \epsilon_S(h_S) - 1 < \epsilon_S(h_S)$$

But this implies that $\epsilon_S(h_S) > \epsilon_S(h_{S^{|i}})$, which means the former does not minimize empirical error. So by contradiction, $h_{S^{|i}}(x_i) = y_i$ implies $h_S(x_i) = y_i$, and $\hat{R}_{LOO}(h_S) \geq \hat{R}_S(h_S)$.

Now, recall that

$$\mathbb{P}\left(|R(h_S) - \hat{R}_S(h_S)| \leq 2\sqrt{\frac{d(\ln(m/d)+1) + \ln(9/\delta)}{m}}\right) \geq 1 - \delta$$

Let $\beta > 0$ be arbitrary. Then with $\mathbb{P} \geq 1 - \beta_1$,

$$R(h_{S^{|i}}) \leq \min_{h \in H}\{R(h)\} + 4\sqrt{\frac{d(\ln(2(m-1)/d)+1) + \ln(9/(\beta_1/2))}{m-1}}$$

$$R(h_S) \leq \min_{h \in H}\{R(h)\} + 4\sqrt{\frac{d(\ln(2(m)/d)+1) + \ln(9/(\beta_1/2))}{m}}$$

On the other hand, with $\mathbb{P} = 1$,

$$R(h_{S^{|i}}) \geq \min_{h \in H}\{R(h)\}$$
$$R(h_S) \geq \min_{h \in H}\{R(h)\}$$

Then with $\mathbb{P} \geq 1 - \beta_1$,

$$R(h_{S^{|i}}), R(h_S) \in \left(\min_{h \in H}\{R(h)\}, \min_{h \in H}\{R(h)\} + 4\sqrt{\frac{d(\ln(2(m-1)/d) + 1) + \ln(9/(\beta_1/2))}{m-1}} \right)$$

That is,

$$|R(h_{S^{|i}}) - R(h_S)| \leq 4\sqrt{\frac{d(\ln(2(m-1)/d) + 1) + \ln(9/(\beta_1/2))}{m-1}}$$

Now set $\beta_1 = \frac{2d}{m}$. Combining the fact that $\hat{R}_{LOO}(h_S) \geq \hat{R}_S(h_S)$ and Theorem 5 we get

$$\mathbb{P}\left(|\hat{R}_{LOO}(h_S) - R(h_S)| \leq \left(8\sqrt{\frac{(d+1)(\ln(9m/d) + 2)}{m}} \right) / \delta \right) \geq 1 - \delta$$

□

As we saw from Theorem 4 and 6, even with stability as a tool, we are not likely to derive a better generalization bound in terms of LOOCV than empirical stability. Nevertheless, empirically speaking, LOOCV usually performs better than empirical error. In order to understand why this happens, we have to classify algorithms as "stable" and "unstable", in hopes to explain why LOOCV empirically indeed performs better on many algorithms.

3 Stability with General Loss Function

3.1 Generalized Hypothesis and Error Stability

In the previous section we have been assuming $0-1$ loss for all our algorithms, and the notions of stability has been defined accordingly. With a general loss function, it is still possible for us to define the two concepts in a similar sense. Instead of using the parameter β_1, we will capture the stochastic nature of sampling by taking the expectation of the distance. [4]

Definition 6. *An algorithm has* ***(generalized) hypothesis stability*** β *with respect to the loss function* l *if* $\forall i \in [1, m]$,

$$\mathbb{E}_{S,(x,y)}[|l(h_S, x) - l(h_{S^{|i}}, x)|] \leq \beta$$

If we restrict $x \in X$ to $x \in S$, we call this property ***pointwise hypothesis stability***

Definition 7. *An algorithm has* ***(generalized) error stability*** *β with respect to the loss function l if $\forall S \sim D^m, i \in [1, m]$,*

$$|\mathbb{E}_{(x,y)}[l(h_S, x)|] - \mathbb{E}_{(x,y)}[l(h_{S^{|i}}, x)]| \leq \beta$$

In this generalized definition, error stability is also a weaker form of hypothesis stability, as we'll independently prove here.

Theorem 7. *If an algorithm is hypothesis stable, it is also error stable with respect to the same loss function.* ***Stable*** *means that $\lim_{m\to\infty} m\beta < \infty$, that is, $\beta = O(\frac{1}{m})$ at least.*

Proof. Suppose the algorithm is hypothesis stable, then

$$\lim_{m\to\infty} m\mathbb{E}_{S,x}[|l(h_S, x) - l(h_{S^{|m}}, x)|] = 0$$

Since S and x are sampled independently, if we further assume the loss functions are integrable (integral on all space is finite), then by Fubini-Tonelli Theorem, we could change the order of integration (hence the order of expectation).

$$\lim_{m\to\infty} m\mathbb{E}_S[\mathbb{E}_x[|l(h_S, x) - l(h_{S^{|m}}, x)|]] = 0$$

Then since the expectation over x is a nonnegative random variable, by Markov's Inequality, $\forall \epsilon > 0$

$$\begin{aligned}\lim_{m\to\infty} m\mathbb{P}_S(\mathbb{E}_x[|l(h_S, x) - l(h_{S^{|m}}, x)|] \geq \epsilon) &\leq \lim_{m\to\infty} m\frac{\mathbb{E}_S[\mathbb{E}_x[|l(h_S, x) - l(h_{S^{|m}}, x)|]]}{\epsilon} \\ &= 0\end{aligned}$$

□

3.2 Uniform Stability

We now introduce a new, stronger notion of stability [4]

Definition 8. *An algorithm has* ***uniform stability*** *β with respect to the loss function l if $\forall S \sim D^m, i \in [1, m]$,*

$$\|l(h_S, \cdot) - l(h_{S^{|i}}, \cdot)\|_\infty \leq \beta$$

We could see why this stronger notion of stability motivates a generalized notion of hypothesis and error stability with respect to a general loss function. If we were to take the $0-1$ loss, the definition would be vacuous. The left hand

side would take value 0 if the two loss functions coincide. If they disagree at any point, there would be a disparity of $l = 0$ versus $l = 1$, making the ∞-norm distance 1. So under the 0-1 loss, the algorithm will be uniformly stable if and only if $f \in H$, in which case $\beta = 0$ is always eventually attained at $m \to \infty$ (so long as the population pool is at most countably infinite, which should be the case in computing due to representing $\mathbb{R}$ with float point numbers), but of course it's a bad idea.

Again, we now prove independently that uniform stability is stronger than hypothesis stability.

Theorem 8. *If an algorithm has uniformly stability β, it also has hypothesis stability β with respect to the same loss function.*

Proof. Suppose the algorithm has uniform stability β. Then $\forall S \sim D^m$,

$$\begin{aligned}
\beta &\geq \|l(h_S, x) - l(h_{S^{|i}}, x)\|_\infty \\
&\geq \sup_{x \sim D} |l(h_S, x) - l(h_{S^{|i}}, x)| \\
&\geq \mathbb{E}_x[|l(h_S, x) - l(h_{S^{|i}}, x)|] \\
&\geq \sup_{S \sim D} \mathbb{E}_x[|l(h_S, x) - l(h_{S^{|i}}, x)|] \\
&\geq \mathbb{E}_S[\mathbb{E}_x[|l(h_S, x) - l(h_{S^{|i}}, x)|]]
\end{aligned}$$

Again, by the integrability of the loss function and the independence of sample drawn, we can apply Fubini-Tonelli Theorem to combine the integration.

$$\mathbb{E}_{S,x}[|l(h_S, x) - l(h_{S^{|i}}, x)|] \leq \beta$$

□

3.3 Generalization Bounds with (Generalized) Stability

Theorem 9. *For an algorithm with pointwise hypothesis stability β with respect to a loss function l bounded by M, $\forall \delta \in (0, 1)$*

$$\mathbb{P}\left(R(h_S) \leq \hat{R}_S(h_S) + \sqrt{\frac{M^2 + 12Mm\beta}{2m\delta}}\right) \geq 1 - \delta$$

if we replace pointwise hypothesis stability with hypothesis stability, we could use LOOCV

$$\mathbb{P}\left(R(h_S) \leq \hat{R}_{LOO}(h_S) + \sqrt{\frac{M^2 + 6Mm\beta}{2m\delta}}\right) \geq 1 - \delta$$

Theorem 10. *For an algorithm with uniform stability β with respect to a loss function l bounded by M, $\forall \delta \in (0, 1)$*

$$\mathbb{P}\left(R(h_S) \leq \hat{R}_S(h_S) + 2\beta + (4m\beta + M)\sqrt{\frac{\ln(1/\delta)}{2m}}\right) \geq 1 - \delta$$

$$\mathbb{P}\left(R(h_S) \leq \hat{R}_{LOO}(h_S) + \beta + (4m\beta + M)\sqrt{\frac{\ln(1/\delta)}{2m}}\right) \geq 1 - \delta$$

If we further assume the algorithm is uniformly stable ($\beta = O(\frac{1}{m})$), the bounds are tight.

Now we have seen that for a uniformly stable algorithm, LOOCV indeed has a better (and tight!) bound than empirical estimates.

4 Practical Example

4.1 Problem

We apply LOOCV estimation to the famous problem on Kaggle of predicting survivorship on the vessel Titanic. The features consists of passenger's sex, age, fare, passenger class, family size, etc. Using these features, we must predict whether that passenger survived or not in the incident. We borrow the dataset created by community contributor AZEEM BOOTWALA, who sanitized invalid data and normalized the features to be directly usable in learning. Since it is a binary classification problem, it is suitable for us to utilize a Support Vector Machine model, where 1 denotes survived and 0 otherwise. The result is as follows:

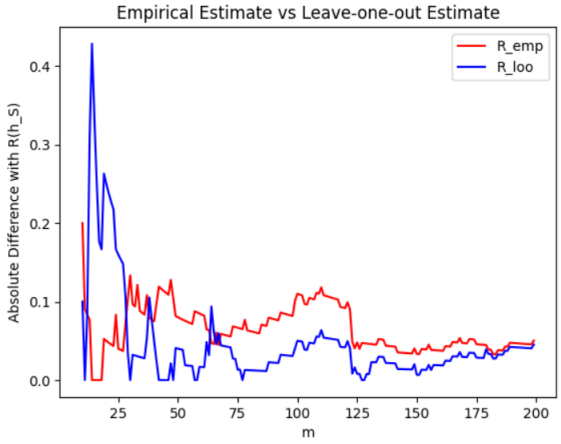

Figure 1: Emp vs Loo for SVM

Algebraically speaking, when m is small, β is large, so the accuracy of both $\hat{R}_S$ and $\hat{R}_{LOO}$ are bad. When $m \to \infty$, $\beta \to 0$, so the difference between the two predictions becomes infinitesimal. It is when m that is moderately large (approximately between 25 and 175) that we see a significant difference take place. This can also be explained using mathematical intuition. At first, when

m is too small, removing or replacing a sample point has a major effect on the model, and $\hat{R}_{LOO}$, trying to take an average across multiple completely different $h_{S^{|i}}$ models, would have a bad accuracy. When m is too large, removing or replacing a sample point has virtually no effect on the model ($h_S \approx h_{S^{|i}}$), and $\hat{R}_{LOO}$, taking an average across these virtually equivalent $h_{S^{|i}}$, will not give a much different result from simply empirically estimating h_S.

If we use an algorithm known to be unstable, then the previous bounds and relationships do not hold.

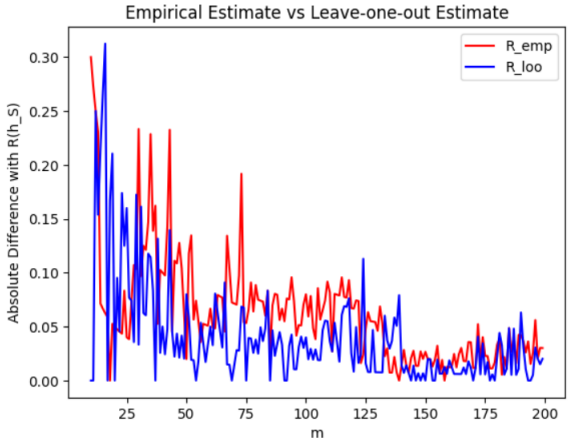

Figure 2: Emp vs Loo for Decision Tree

5 Stability (and Unstability) of Algorithms

With the above bounds present, the problem now shifts to understanding what algorithms are stable, and what are not, and if they are stable, how do we bound its stability constant β. One important intuition in our discussion is that with the proof of Theorem 6, we have proven a lemma where $\hat{R}_{LOO} \geq \hat{R}_S$ for all empirical error minimizing algorithms. So if $\hat{R}_S$ overestimate the generalization error, $\hat{R}_{LOO}$ is always a worse estimate. This is to say that there's generally no hope in proving the superiority of $\hat{R}_{LOO}$ for error-minimizing algorithms. We should then focus our attention only on algorithms with regularization terms present.

We have seen earlier that the naive $0-1$ loss for binary classifier cannot induce stability in a non-trivial way. We now introduce a new loss function

$$l_\gamma(y, y') = \begin{cases} 1 & \text{if } yy' < 0 \\ 1 - yy'/\gamma & \text{if } 0 \leq yy' \leq 1 \\ 0 & \text{if } yy' > 1 \end{cases}$$

With respect to this loss, the k-NN algorithm is unformly stable with stability[4]

$$\beta \leq \frac{k}{m}$$

The same paper pointed out that in a Hilbert space with a loss function $l(\cdot, \cdot)$ induced by a inner product $\langle \cdot, \cdot \rangle$ whose induced norm $\| \cdot \|$is bounded by κ, and a k-norm regularization with λ multiplier, SVM regression is uniformly stable with

$$\beta \leq \frac{\kappa^2}{2\lambda m}$$

and soft margin SVM binary classification is uniformly stable with also

$$\beta \leq \frac{\kappa^2}{2\lambda m}$$

On the other hand, decision tree is known to be an unstable classifier. Its instability comes from the fact that if a data point is removed and a data split is changed, it will affect the entire subtree stemming from the split [5]. Another geometric intepretation is to treat a decision tree as a partition scheme of a linear space into hyper-rectangles. If one of those hyper-rectangles are changed, so must other hyper-rectangles that are adjacent to it. The same paper described a split selection algorithm to strategically choose the split that affects stability the least, which alleviates but not eliminate the problem.

Analyzing the stability of neural networks proves to be a difficult task, with few literature addressing the topic without imposing harsh specific conditions that impede generality. Hardt et al. [7] sheds some light on the issue by analyzing the stability of stochastic gradient descent, a method of optimization popularly used for neural network's objective function.

Theorem 11. *Suppose the loss function is locally α-differentiable, convex, and L-lipschitz at the minimizer, and the algorithm is equipped with stochastic gradient descent with step size (learning rate) $r_t \leq 2/\alpha$ and T steps, then the algorithm is uniformly stable with stability*

$$\beta \leq \frac{2L^2 \sum_{t=1}^{T} r_t}{m}$$

More recently, some scholars argue that uniform stability is still too weak a notion to deal with neural network related problems because of its independence from data labeling, which leads to pessimistic bounds with many restrictions. They call for a stronger notion that could distinguish models trained on the true labels (small generalization error) and models trained on the random labels (large generalization error) [8].

References

[1] V.N.Vapnik., Estimation of Dependences Based on Empirical Data. Springer-Verlag, New York, 1982. pp. 160.

[2] M. Kearns and D. Ron, Algorithmic stability and sanity-check bounds for leave-one-out cross-validation, Neural Comput. 11(6) (1999) 1427–1453.

[3] L. P. Devroye and T. J. Wagner. Distribution-free performance bounds for potential function rules. IEEE Transactions on Information Theory, IT–25(5):601–604, 1979.

[4] O. Bousquet and A. Elisseeff. Stability and generalization. J. Mach. Learn. Res., 2:499–526, 2002.

[5] Ruey-Hsia Li and Geneva G. Belford. 2002. Instability of decision tree classification algorithms. https://dl.acm.org/doi/10.1145/775047.775131

[6] Saurabh Verma, Zhi-Li Zhang, "Stability and Generalization of Graph Convolutional Neural Networks". https://doi.org/10.48550/arXiv.1905.01004

[7] Moritz Hardt, et al. "https://arxiv.org/pdf/1509.01240.pdf". https://arxiv.org/abs/1509.01240.

[8] Chiyuan Zhang, et al., "Understanding deep learning (still) requires rethinking generalization". https://dl.acm.org/doi/fullHtml/10.1145/3446776/