Journal of Visualization and Interaction

PAC Learning Or: Why We Should (and Shouldn't) Trust Machine Learning

Dylan Cashman

A peer-reviewed publication in the Journal of Visualization and Interaction (JoVI).

This article is an extended version of a publication from the VISxAI 2023 workshop.

DOI: 10.54337/jovi.v1i1.11205 (the public peer reviews can also be found at this link)

- Submitted: 2023-07-29
- Accepted: 2025-09-23
- Published: 2025-12-10 (version 1)

Publisher

Aalborg University Open Publishing
Kroghstræde 1-3
9220 Aalborg Øst
DENMARK
ISSN: 2794-5502

PAC Learning

Or: Why We Should (and Shouldn't) Trust Machine Learning

By: Dylan Cashman, Assistant Professor of Computer Science, Brandeis University
October 3, 2025

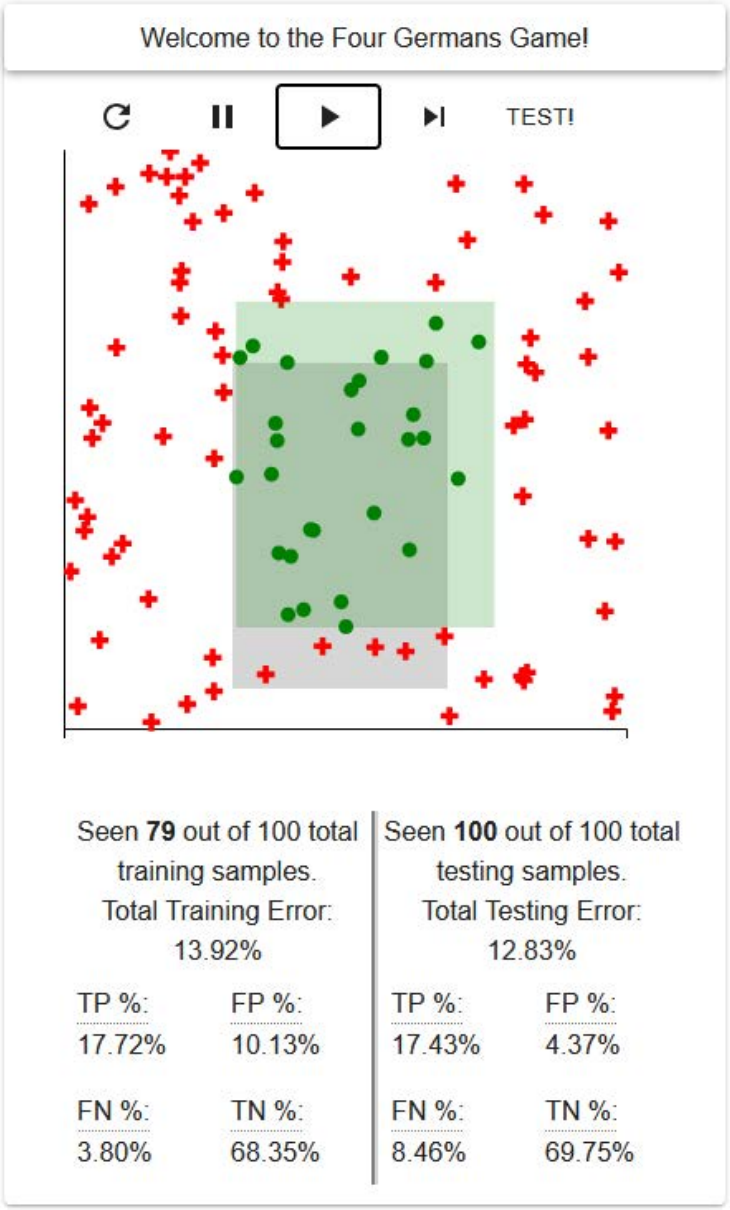


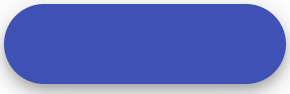
Table of Contents

Abstract

In this interactive article, we present an interactive game that represents the types of tasks solved by machine learning algorithms. We use this game to motivate the definition of Probably Approximately Correct (PAC) learning, illustrating a proof of PAC learnability for Empirical Risk Minimization (ERM). Then, we show three types of vulnerabilities of ERM that often occur in applied machine learning - domain mismatch, dependencies in data, and an incorrect model class. We conclude by arguing for the need for visualization to identify these issues in a modeling dataset.

Source code and any associated materials can be found on OSF.

Introduction



Like programming your microwave clock or cutting open a pineapple, learning about machine learning is one of those things that seems intimidating, but ultimately is pretty logical. I'm going to illustrate to you a few basic principles of machine learning using a simple interactive game, which you will play using the interface on the right side of your screen.

In this game, you are trying to guess a rectangular region, called a **concept**, that represents some subset of the universe. The cartesian plane, what we think of as a pair of perpendicular x and y axes, represents the space of all possible x and y values, and the concept you are trying to figure out is a subset of this x and y space. One way to think of the game is that if the x represented weight, and the y axis represented height, I might ask you to guess what rectangular region represents the concept of "men of medium height and weight".

I don't give you any labels on your x and y axes; instead, I give you a set of examples that are in the concept (which we will refer to as having a **true** label), and a set of examples that are outside of the concept (which we will refer to as having a **false** label). You use the labeled data that I provide for you to guess what is the definition of the concept that data represents. This simple task is actually very similar to the task a machine learning algorithm does to build a model, whether it be a facial recognition model, a weather prediction model, or even a large language model. When

provided with some data to learn from (typically called a **training set**), the machine learning algorithm applies a strategy to guess a concept. Then, the model representing that concept can be tested out on some held-out or real-world data, and hopefully, it performs pretty well.

Before proceeding to the next section, I want you pretend to be a machine learning algorithm, using the interface to your right. On the x and y plane, you can click and drag a rectangular region to represent your guess at the concept. By hitting the TEST! button, additional held-out data will be evaluated against your concept. Your guessed concept will be evaluated based on what percent of randomly drawn points would be True Positives (TP) and True Negatives (TN) - i.e. new points that are correctly classified by your guessed concept - vs. False Positives (FP) and False Negatives (FN) - i.e. new points that are incorrectly classified by your guessed concept. For more details on these metrics, mouse over them in the interactive display on the right side of the page.

Once you've tested out your concept, play again by using the ↻ button to start a new game. Each time you start a new game, the browser will guess a new random rectangular region for you to guess. When you start a new game, you will note that training data doesn't show itself all at once - it trickles in slowly. You can change the speed with which the data trickles in by either pausing it with ⏸, unpausing it with ▶, or changing it to double speed with ⏭. Try to challenge yourself by pausing the game after only 10 or 20 points and guessing the concept those points represent. You'll find that in some cases, it's easy if you are lucky on which points are provided to you in those first 10 or 20 points, but in other cases, it's almost impossible. When you have few points, all you can do is select a concept that is consistent with the points you've seen, i.e. your training set - and your concept will typically have great performance on the points in your training set. But you may have high error on your testing set.

I first heard of this game from Tufts University professor emeritus Anselm Blumer, who along with his coauthors Ehrenfeucht, Haussler, and Warmuth used this setting to demonstrate the concept of learnability in their article in 1989. That paper became known as the "Four Germans Paper", and in honor of their work, I call this game the *Four Germans Game*. This game is a useful abstraction that can help illustrate the task that machine learning algorithms have to solve. Machine learning models see some training data generated by an underlying phenomenon (i.e. the hidden box), and based on that limited information must fit a model that approximates the underlying phenomenon. A successful algorithm will pick a concept with low error on its training data, and will hopefully have similarly low error when it is tested (here error is defined as the total area that is only inside one box, i.e. false positives or false negatives). A lot can go wrong, however, even in this simple game:

- What if the training data is not representative of the actual phenomenon?
- What if there is not enough training data?
- What if the training data is noisy?

- What if the wrong model is used (i.e. you guess a rectangular region, but the actual ground truth is a circle)?

This game can be used to illustrate many concepts of one of the underlying theories of machine learning: probably approximately correct (PAC) learning, proposed by Leslie Valiant in 1984. PAC learning is a framework to help understand why we are able to trust machine learning algorithms. It also provides some insight into why machine learning models can make nonsensical and erroneous predictions. In this article, we'll illustrate the basic assumptions of most machine learning algorithms, and demonstrate why fully-automated approaches may never be able to be error-free if these assumptions are validated.

Why this is important

Machine learning algorithms are becoming ubiquitous tools in the age of big data to aid in decision making and data analysis. They can be deployed in complex situations where the data they are acting on has different qualities than the data they were training on. A common example is e-mail spam filters: the text of many spam emails was used to train spam filters. However, since the strategies used by e-mail spammers evolve over time, the training dataset might not be representative of newer spam emails, so those emails get through. This is a manifestation of the *domain mismatch* vulnerability illustrated in this article. Understanding the ways in which machine learning algorithms can have worse performance than we expect is important to designing more resilient algorithms.

This is increasingly important as machine learning models are responsible for broader use cases. They are actively being deployed in scenarios where their impact directly effects humans, including self-driving cars, patient triaging software at hospitals, and even crime prediction software, a la Minority Report. Automation from deep learning models such as facial recognition technology, image and scene classification, and text generation through large language models are rapidly eclipsing levels of human competency and changing the way we live our lives.

When they are successful, these models can result in new industries and capabilities that were science fiction only years ago. But when they fail, they can have drastic consequences for the humans affected by them. Consider a scenario where making a mistake can have drastic consequences: for example, a model might be used to prescribe a medication that has drastic side effects if applied to the wrong patient. Would you feel comfortable using a machine learning model to decide on that medication? When we deploy machine learning algorithms out in the wild, how do we *know* that they will work at their intended goal? How do we know that the probability of their error is small *enough*?

The answer, of course, is that we don't know for sure the models will work as intended. We train models on an available dataset, and then hope that the model's performance on a portion of that dataset is representative on however that model will be used out in the wild. If a model is deployed

out in the world in a way that it will face very different data than the data it is trained on, then it could perform much worse than it was observed performing during training.

It is important for the general public to understand that machine learning algorithms make mistakes. And it is *crucial* for decision makers to understand *why* machine learning models can perform worse when deployed in the real world, so that they can make informed decisions about deploying these models in high-risk environments. In hindsight, it might seem obvious that the training data of facial recognition algorithms could lead to a bias in a deployed system. PAC learning, however, will provide us foresight to anticipate these types of risks before building a model. And this article will illustrate what that means.

Four Germans Game: Find My Rectangle

Let's return to the game, and consider some machine learning concepts. First, try to play the game again (in the smaller version to the right here), but without getting any training data.

...

Not very fun, or interesting, right?

This is an illustration of there being No Free Lunch theorems for optimization. As described by David Wolpert, a researcher in machine learning theory and author of several No Free Lunch (NFL) papers, *"the NFL theorems say that under a uniform distribution over problems (be they supervised learning problems or search problems), all algorithms perform equally."* In the absence of any information, we can do no better than random chance, and thus we can make no statements about how "correct" we are. You might have 10% error, you might have 90% error, you might fit the rectangle perfectly and have 0 error.

However, suppose we are given some information. Here, we are given 10 points, with 5 inside the target rectangle and 5 outside. If you guess a rectangle, can you make any guarantees about how close you are to the ground truth?

Consider the various strategies that you might use to minimize your error. Do you tightly wrap the green examples, or do you leave some space around them to allow for data you haven't seen yet?

Which strategy generally works better? Which strategy works better in the unlucky case, where the sampled data doesn't provide much information due to bad luck?

Up to this point, you have been playing the machine learning algorithm. You have all the magnificent faculties of the human mind, letting you change and adapt your strategy as you encounter new examples. But machine learning algorithms tend to be much simpler, and typically more stuck-in-their ways, because they have to be well-defined.

We present three simple machine learning algorithms to solve this problem.

TIGHTEST FIT

1. **Tightest Fit**: In this algorithm, the tightest possible rectangle is chosen that still contains all of the positive examples seen so far. This algorithm is the smallest possible region that is still consistent with the samples seen. It is related to the principle of Risk Minimization, which is a strategy applicable to many machine learning problems. This will under-estimate the rectangular region.

LOOSEST FIT

1. **Loosest Fit**: In contrast to the Tightest Fit algorithm, this algorithm chooses the loosest possible rectangle. This rectangle will be internal to all negative examples, and will contain all positive examples. This will over-estimate the rectangular region.

MAXIMUM MARGIN

1. **Maximum Margin**: This is a midpoint between the two previous algorithms. It chooses a rectangle that is as far as possible from both the positive and negative examples, while containing all examples.

How do we determine if any of these algorithms are good? Or reliable? What are the properties and assumptions that are necessary to make these judgements?

PAC-learning

When we use classical statistical methods, such as logistic regression, to analyze data, we can *trust* the model works because formalisms such as the p value and confidence intervals give us bounds on the possible errors of the model. The rectangle game proposed by Blumer et al. that is featured in this post is designed to demonstrate how such formalisms can be defined for machine learning techniques.

Computational Learning Theorists attempt to find theoretically sound definitions of concepts found in machine learning, such as training data, validation accuracy, and modelling. These definitions are then used to prove bounds on various metrics like error and runtime. The paradigm of **probably approximately correct learning**, or **PAC learning**, has the explicit goal of determining under what conditions a machine learning algorithm will most likely perform about as well as it does on a training set, when deployed in the wild.

We provide the following treatment based on chapter 1 from Kearns and Vazirani, and we direct the reader to that text for a more thorough discussion. Loosely, a concept is PAC-learnable if there exists a machine learning algorithm such that, after viewing a sufficient amount of labeled samples, we can prove a bound on the error (let's say that error is smaller than some epsilon ϵ), assuming we aren't too unlucky with which samples we receive (so, with probability at least $(1 - \delta)$). This type of analysis mirrors analysis done using p values for logistic regression or mathematical definitions of limits, and allows us to bound error on machine learning models. We define it formally below, then prove that our game is PAC-learnable. The assumptions in the definition and its proof lead us to a better understanding of potential vulnerabilities of machine learning algorithms, which will be demonstrated below.

PAUSE THE GAME

Definition

(Adapted from Wikipedia)

Let X be a data space. This could be as simple as our 2-D space, or it could be the space of all images or the space of all snippets of text. A *concept* is a subset $c \in X$ of the data space - in

our case a rectangular region. A *concept class* C is the collection of all concepts over our data space - in our case, all possible rectangular regions in the 2-D space.

Let $EX(c, D)$ be a procedure that draws an example, x, using a probability distribution D and gives the correct label $c(x)$. In our case, $EX(c, D)$ is the process that happens when you hit the ▶ button. It draws a random location in the 2-D space using D, the uniform distribution, then provides a label for that location as green (if it is part of the rectangular region we want to guess), or red (if it is outside that region).

An algorithm A is a **PAC learning algorithm** for C (equivalently, the concept space C is **PAC learnable**) if the following is true: For any ϵ and δ that we choose between 0 and 1 (i.e. some small fraction of a number), there exists a finite sample size p such that if $EX(c, D)$ draws and labels p examples, the algorithm A will output a hypothesis concept, $h \in C$, that will have less than ϵ error, with probability $1 - \delta$.

In other words, a type of concept (which could be rectangular regions on a plane, or the set of all pictures of cats and dogs within a group of images - the thing we are trying to see in the data space) is considered learnable if a learning algorithm exists that can usually reach a level of guaranteed performance if it sees enough training data. In the definition, choosing ϵ chooses how "good" an algorithm we want. If $\epsilon = 0.05$, then we are saying a "good" algorithm will have less than 5% error, however we want to define error. The δ instead compensates for bad luck. If $\delta = 0.1$, then we are saying that we will be able to produce a "good" algorithm at least 90% of the time. The other 10% of the time, we might be unable to produce a "good" enough algorithm because of getting unlucky with a sample size. In general, the number of samples increases as the amount of error (ϵ) and bad luck (δ) we tolerate goes down. These two parameters in the definition explain the redundancy in the term: probably (δ) approximately (ϵ) correct.

It's useful to consider when a concept is *not* PAC learnable. It would seem pretty universal that the more samples a learning algorithm sees, the more confident it would get about its predictions, and the better those predictions would be. But in a lot of problems, you hit a plateau. If you try to fit a very simple model, like a logistic regression classifier, to a very complicated concept, like classifying photos of dogs, you probably can't guarantee any arbitrary level of performance no matter how many samples you see.

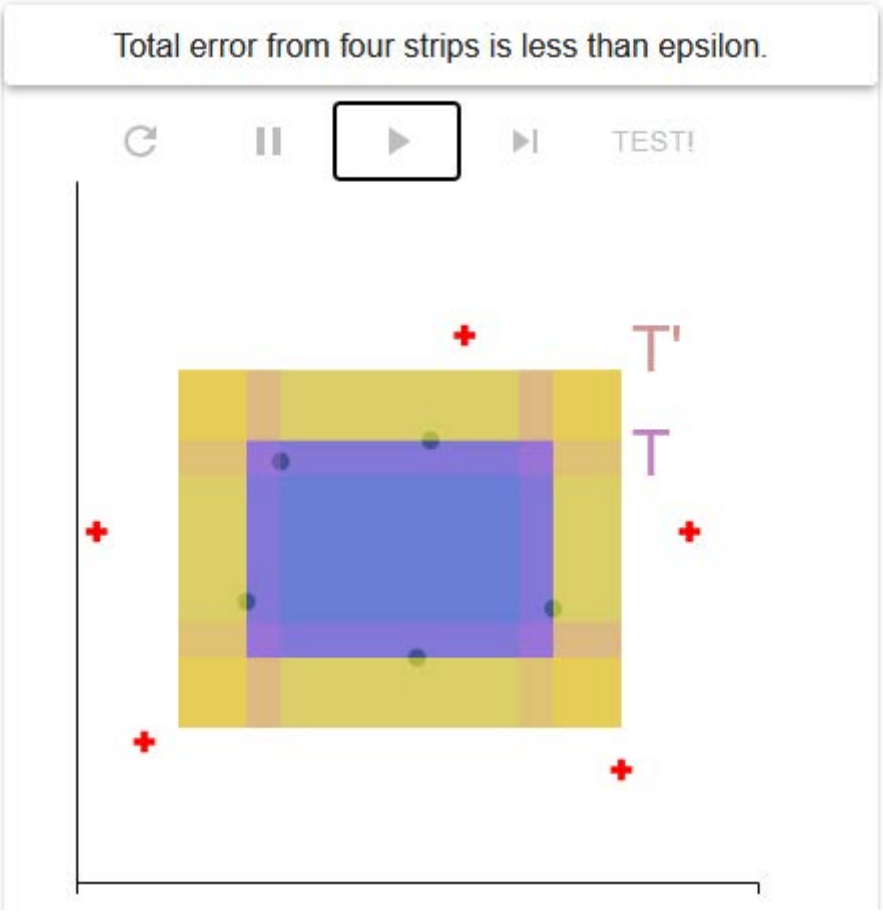
Why Should We Care About PAC Learnability

The most popular learning paradigm used in machine learning use cases is Empirical Risk Minimization, which says that when we want to minimize error (and thus risk) when putting a machine learning model out in the world, we should first tune it to minimize error on the *empirical* data that we have, i.e. the training data. By minimizing the *empirical risk*, we are hopefully minimizing the *actual risk* the model will try to avoid when it is used out in the world.

This only makes sense, though, if there is some theoretical basis for why seeing labeled training data would guarantee performance on unseen data. If a concept class wasn't PAC learnable, then no matter how good our performance on a training set was, we wouldn't be able to make any guarantees on an algorithm's performance out in the world.

Understanding how to prove PAC learnability can help us see what stipulations about a machine learning use case need to hold in order to trust empirical risk minimization. The definition of PAC learnability can also provide hints into how applied machine learning can go wrong.

Now, we will use our game to demonstrate first how a learning algorithm can be proven to be a PAC learning algorithm. Then, we will highlight different ways where PAC learning can break down due to issues with the data or the complexity of the concept that we are trying to model.



TIGHTEST FIT

We can use the PAC learning paradigm to analyze our algorithms. In particular, we can look at the tightest fit algorithm, and try to bound its error, where error is defined as the probability that our

algorithm's chosen region will get the label incorrect. We would like to prove that the error is less than ϵ, with probability at least $1 - \delta$. We do this by showing that, under the uniform distribution that gives us our labeled points, we have a pretty good chance of having low error.

TIGHTEST FIT ERRORS

First, we note that the tightest-fit rectangle is always contained in the target rectangle. We can express the total error region as four strips: the strips above, below, to the left, and to the right of the tightest-fit rectangle. This is the only region where the model chosen by our algorithm will label incorrectly. To proceed, we would like it if the size of each region is $\leq \frac{\epsilon}{4}$, so that the total size of the error regions is less than ϵ.

We want to calculate the probability that our model makes mistakes. Since this only happens in the error region, if the data is uniformly distributed, we can count how many random samples we would generally need to receive labels for before we would be pretty sure that we would receive points in these error regions. This will give us a relationship between ϵ and δ in terms of the number of samples drawn, M.

ERROR FROM TOP STRIP

Consider the top strip, which we'll call T'. Then, consider the strip T that starts from the top of the proposal region, and sweeps down in height until it has area $\frac{\epsilon}{4}$. **If T' is contained in T, then it has area less than $\frac{\epsilon}{4}$, and we are done**. By our algorithm, we know that our T' can only be greater in area than T if we haven't seen any samples in T, since if we had, then our proposal region would contain that point. The probability that M independent draws from the distribution (i.e. new points that we see) all miss the region T (and thus that the area is less than $\frac{\epsilon}{4}$) is exactly $(1 - \frac{\epsilon}{4})^M$.

ERROR FROM ALL STRIPS

The same analysis holds for the other three regions - the probability that *any one* of the four regions (above, below, to the left, to the right) have area $> \frac{\epsilon}{4}$ is exactly $(1 - \frac{\epsilon}{4})^M$. And, if *all* of them have area $\leq \frac{\epsilon}{4}$, then we have bounded our total error region to be $\leq \epsilon$. We need to calculate the probability that this is not true - that any of these regions is too large. By Boole's Inequality, the probability that at least one of the four strips have area $> \frac{\epsilon}{4}$ is $\leq 4(1 - \frac{\epsilon}{4})^M$.

While this may all seem meandering, we are actually almost there. What we've shown is that the probability that our model has sufficiently small error ($\leq \epsilon$) is dependent on M, the number of samples we've seen. In this game, we get to wait, and let M increase - we get to wait to see more labeled points before we choose our tightest fit rectangle. To get our *probably approximately correct* model, we choose M to satisfy $4(1 - \frac{\epsilon}{4})^M \leq \delta$, i.e. the probability that the area is **not** bounded by ϵ is smaller than δ, so the probability that it **is** bounded by ϵ is greater than $1 - \delta$.

The final step of the proof is to determine how many samples we'd need - solving for M. Using the inequality $(1 - x) \leq e^{-x}$, we can simplify the bound to $4e^{\epsilon M/4} \leq \delta$, and solve for $M \geq (4/\epsilon)ln(4/\delta)$. Switching out bounds for more mathematically convenient ones is a very common strategy for establishing bounds in these types of proofs. The key takeaway is that M is not infinity. Since M is a real, finite number, we've demonstrated that **the problem is PAC-learnable**, i.e. we have bounded the error in terms of the number of samples seen.

Assuming the Worst

As the old saying goes: Don't Assume. It makes an ass out of you and me. And our proof involved many assumptions that may not be so realistic, if we imagine proving something similar for a more applied case.

While the proof contains some complicated logic, and our conclusion can be hard to interpret, it was actually a significant result. For this very simple problem (much simpler than the types of problems machine learning is being used to solve in the real world), we were able to show that the error on unseen examples would be bounded.

But not so fast - there's a major issue that we didn't mention. The proof we went through, as well as the PAC-learning paradigm in general, relies on several assumptions.

Independent and Identically Distributed Data

NOT IID DATA

That the sampled data in independent and identically distributed (typically referred to as i.i.d). In real world data, this may not be the case; if the algorithm is being trained on streamed data, it typically cannot be considered independent draws. As a more practical example, suppose our training data was patient visits to a doctors office, in chronological order. This data would have some sort of seasonality to it - the visits in the winter would typically involve fevers and respiratory problems, while the visits in the summer would involve more sports-related accidents. This would violate the i.i.d. assumption because we wouldn't be able to know that the training points we had

seen would have bearing on the training points we were going to see in the future. This assumption was used in our probability bounds.

In this version of the game, the i.i.d. assumption is violated by placing some linear drift on the data. As we get new points, they move linearly in a certain direction.

Domain Mismatch

DOMAIN MISMATCH

We assume (or at least count on) that the data that we test our algorithm on is from the same distribution that we train our algorithm on. This assumption is frequently incorrect in practice, and can result in unexpected behavior by the machine learning algorithm. This phenomenon is called *domain mismatch*, or sometimes *train test mismatch*, because the domain of the data that the model is trained on is not the same domain of data that the model will be deployed or evaluated on. Consider a computer vision algorithm for a self-driving car that only trained in the Pacific Northwestern United States, and then was deployed in a desert climate; the behavior of the computer vision algorithm in training might have no relationship with its behavior in testing. This would correspond to the testing samples being labeled in a different region than during training.

In this version of the game, the training distribution and the testing distribution are independent of one another. This means that the true labels (green points) that you get in training may not be representative of where the true labels would appear in the testing phase. It makes it impossible to guarantee any performance: it's back to the No Free Lunch situation described earlier.

Incorrect Model Class

ELLIPSES, NOT RECTANGLES

Our proof assumed that we already knew that the type of region we were looking for was a rectangle. But in practice, we rarely know what kind of machine learning model will match the phenomenon being modeled in the data. Suppose, instead of a rectangular region, the region we were looking for turned out to be an ellipse, or a parallellogram, or even an arbitrary, amorphous disconnected region. A geometric proof would then be much harder, or even impossible. The field of adversarial machine learning gives us practical examples. For example, small amounts of noise added in to images that change the label of images take advantage of the high-dimensional decision boundaries found in convolutional neural nets, as in Tabacof and Valle (2016).

In this version of the game, the actual region we are trying to guess is an ellipse, but we can only choose a rectangle. Play it a few times and notice that no matter how good your guess is, you will always have some nonzero lower bound on your testing error.

There are other considerations for failure scenarios for machine learning algorithms. Varshney presented an interesting treatment in 2017. In practice, there is active research into proving PAC learnability for more complex learning spaces than in this article (PAC Learnability of Node Functions in Networked Dynamical Systems by Adiga et al. in ICML from 2019, or A Theory of PAC Learnability under Transformation Invariances by Shao et al in NeurIPS from 2022). However, it is likely that PAC learnability wouldn't hold in real world cases because we typically can't *prove* (or *know*) that the assumptions are going to be held. But the paradigm of PAC learnability illustrates what types of problems we have to look out for in applied machine learning.

An Argument for Visualization in Applied Machine Learning

The game played in this blog post was hopefully a relevant illustration of some of the topics introduced. If it's helpful, it may be because it is a visual explanation. In general, visualizing the right properties of the data and the algorithm can help indicate to a data scientist whether any of the necessary assumptions are broken. The data scientist can then address them by cleaning or preprocessing the data, or choosing a different machine learning algorithm.

In this blog post, we looked at an incredibly simple machine learning problem, and the algorithms we considered were easily explainable in a sentence or two. Even for this simple problem, it was difficult to prove any limitations on error. Machine learning is typically used to model much more complex problem domains. In these cases, it is very unlikely that error bounds are proveable, and even if they are, it is unlikely that the assumptions are upheld.

Instead, a human in the loop, armed with appropriate visualizations and analytical tools, can act as a safeguard against the most endemic cases. This additional line of defense is more and more necessary as machine learning models are deployed in scenarios that directly affect humans. More discussion on this topic can be found in chapters 1 and 7 of my thesis, Bridging the Human-Machine Gap in Applied Machine Learning with Visual Analytics, which offers additional perspective on the role of visual analytics systems in empirical risk minimization.

Conclusion

In this interactive article, we used the Four Germans Game to illustrate the concept of probably approximately correct (PAC) learning. We then demonstrated how several assumptions are used in

learning theory to build theoretical bounds on performance for machine learning models in the empirical risk minimization learning paradigm. We presented some examples, using the game, of how these assumptions might be broken.

Special Thanks

Thank you to Anselm Blumer and the reviewers from VisXAI and the The Journal of Visualization and Interaction (JOVI).

References

- Blumer, Anselm, et al. “Learnability and the Vapnik-Chervonenkis dimension.” Journal of the ACM (JACM) 36.4 (1989): 929-965.
- Valiant, Leslie G. “A theory of the learnable.” Communications of the ACM 27.11 (1984): 1134-1142.
- Kearns, Michael J., and Umesh Vazirani. An introduction to computational learning theory. MIT press, 1994.
- Feng, Alice, et al. “The myth of the impartial machine.” The Parametric Press (2019).
- Vapnik, Vladimir. “Principles of risk minimization for learning theory.” Advances in neural information processing systems 4 (1991).
- Varshney, Kush R., and Homa Alemzadeh. “On the safety of machine learning: Cyber-physical systems, decision sciences, and data products.” Big data 5.3 (2017): 246-255.
- Valiant, Leslie. “Probably approximately correct: nature’s algorithms for learning and prospering in a complex world.” (2013).
- Adiga, Abhijin, et al. “PAC learnability of node functions in networked dynamical systems.” International Conference on Machine Learning. PMLR, 2019.
- Shao, Han, Omar Montasser, and Avrim Blum. “A theory of pac learnability under transformation invariances.” Advances in Neural Information Processing Systems 35 (2022): 13989-14001.
- Badue, Claudine, et al. “Self-driving cars: A survey.” Expert Systems with Applications 165 (2021): 113816.
- Kwon, Joon-myoung, et al. “Validation of deep-learning-based triage and acuity score using a large national dataset.” PloS one 13.10 (2018): e0205836.
- Wu, Shaobing, et al. “Crime prediction using data mining and machine learning.” The 8th International Conference on Computer Engineering and Networks (CENet2018). Springer International Publishing, 2020.
- Fujiyoshi, Hironobu, Tsubasa Hirakawa, and Takayoshi Yamashita. “Deep learning-based image recognition for autonomous driving.” IATSS research 43.4 (2019): 244-252.

- Buolamwini, Joy, and Timnit Gebru. “Gender shades: Intersectional accuracy disparities in commercial gender classification.” Conference on fairness, accountability and transparency. PMLR, 2018.
- Wolpert, David H., and William G. Macready. “No free lunch theorems for optimization.” IEEE transactions on evolutionary computation 1.1 (1997): 67-82.
- Wolpert, David H. “What is important about the no free lunch theorems?.” Black box optimization, machine learning, and no-free lunch theorems. Cham: Springer International Publishing, 2021. 373-388.
- Tabacof, Pedro, and Eduardo Valle. “Exploring the space of adversarial images.” 2016 international joint conference on neural networks (IJCNN). IEEE, 2016.

Research Material Statements

No datasets are used in this interactive article. The data visualized within the game is either randomly generated or hardcoded in. You can view the source code for this article on OSF. Please see the software license below.

About The Author

Dylan Cashman is an assistant professor in the Michtom School of Computer Science at Brandeis University in Waltham, MA. He previously worked as a Senior Expert in Data Science and Advanced Visual Analytics within the Data Science and Artificial Intelligence division of Novartis Pharmaceuticals in Cambridge, MA. Dylan holds a PhD in Computer Science from Tufts University and an Sc. B in Mathematics from Brown University. His research interests include the development and evaluation of visual affordances that improve usability of artificial intelligence models and data science processes. His research has won best paper awards at the IEEE VIS conference, the Eurovis conference, the Symposium on Visualization for Data Science (VDS), and the Workshop on Visual Analytics for Deep Learning at IEEEVIS.

License

Conflict of Interest

The author declares that there are no competing interests.