Teaching Machine Learning to a Diverse Audience: the Foundation-based Approach

Hsuan-Tien Lin HTLIN@CSIE.NTU.EDU.TW
Department of Computer Science and Information Engineering, National Taiwan University

Malik Madgon-Ismail MAGDON@CS.RPI.EDU
Department of Computer Science, Rensselaer Polytechnic Institute

Yaser S. Abu-Mostafa YASER@CALTECH.EDU
Department of Electrical Engineering and Department of Computer Science, California Institute of Technology

Abstract

How should we teach machine learning to a diverse audience? Is a 'foundation-based' approach appropriate or should we switch to a 'techniques-oriented' approach? We argue that the foundation-based approach is still the way to go.

The foundation-based approach emphasizes the fundamentals first, before moving to algorithms or paradigms. Here, we focus on three key concepts in machine learning that cover the basic understanding, the algorithmic modeling, and the practical tuning. We then demonstrate how more sophisticated techniques like the popular support vector machine can be understood within this framework.

1. Introduction

Machine learning, 'the art of learning from data', is now a wide discipline attracting theorists and practitioners from social media, biology, economics, engineering, *etc.* The discipline represents a prevailing paradigm in many applications: we don't know the solution of our problem, but we have *data* that represents the solution. A consequence is that the typical entry level machine learning class will consist of a diverse audience as seen in most university offerings – multiple backgrounds, motivations and goals. To name a few cases: At the National Taiwan University, the 2011 machine learning class was of size 77 and consists of 42 students from computer science, 26 from electrical engineering, and other students from finance, information management, mechanical engineering and bio-engineering. At Rensselaer Polytechnic Institute, the 2011 machine learning class is of size 50, consisting of 24 undergraduate and 26 graduate students, whose majors are spread over computer science, electrical engineering, information science and engineering, physics, cognitive science, mathematics, chemistry and biology. At California Institute of Technology, the 2012 machine learning class consists of 151 registered students coming from 15 different majors. The online version of the class (`http://work.caltech.edu/telecourse`) is vastly more diverse, having practitioners with minimal background all the way to a winner of the US National Medal of Technology.

Machine learning in many universities is commonly listed as an advanced undergraduate course, relying on some background in calculus, linear algebra and probability. We don't know whether this status will go the way of calculus and introductory computer programming, with separate offerings for biology, management, engineering, *etc.* But we do know that we have to address the diversity challenge facing instructors now. And, perhaps the most important choice to be made is between solid foundations versus practical techniques that can immediately be used. Of course, to keep a student interested, some practical techniques must be there, but where should the focus be?

It's tempting to present the latest and greatest techniques; that is certainly the sexiest way to go. A typical depth-based approach would focus on one or few promising techniques, such as the support vector machine algorithm (Vapnik, 1998). The approach aims at

Appearing in *Proceedings of the Workshop on Teaching Machine Learning*, Edinburgh, Scotland, UK, 2012.

examining the technique thoroughly, from its intuitive and theoretical backbone to its heuristic and practical usage. Students who are taught through this approach can master the particular technique and exploit it appropriately in their domain. However, restriction to one or a few techniques has difficulty satisfying the needs of a diverse audience.

The other extreme is the breadth-based approach. A typical breadth-based approach travels through the forest of learning paradigms and algorithms, jumping from paradigm to paradigm, from algorithm to algorithm. The goal is generally to cram enough together in an attempt to be 'complete'. An ambitious instructor may intend to cover supervised learning (including nearest neighbors, radial basis functions, support vector machines, kernel methods, neural networks, adaptive boosting, decision trees, random forest); unsupervised learning (including principal component analysis and k-means clustering); probabilistic modeling (including Gaussian processes and graphical models); reinforcement learning; *etc.* We can certainly see the appeal of this approach to a diverse audience, but let's not forget that the student has to actually be able to *use* the techniques in their domain. Students can easily get lost in this forest.

> *It is dangerous to have tools available to use without knowing when and how to use them.*

Given the plethora of open source implementations of cutting edge techniques on the Internet. we can always download the latest and greatest open-source implementation of the current cutting edge technique, for example the support vector machine. So it seems that the right place to start is the foundations, *even for a diverse audience*. We can only argue from experience, and we say this on the basis of more than a decade of successful experience[1] in teaching this material.

Foundations First. By foundations we do not mean death by theory. To pull this off, it is necessary to isolate the crucial core that all students should know; the core that will enable them to understand the map of machine learning and navigate through it on their own. We have distilled a three-step core that through experience works well, and we feel that any student of machine learning who has mastered this core is truly poised to succeed in any machine learning application.

[1]Professor Y. Abu-Mostafa received the Caltech Richard P. Feynman prize for excellence in teaching in 1996; Professor Hsuan-Tien Lin received the NTU outstanding teaching award in 2011.

1. *Learnability, approximation and generalization*: when can we learn and what are the tradeoffs?
2. *Careful use of simple models*: the linear model coupled with the nonlinear transform is typically enough for most applications. *Only when it fails should we go after a more complex model.*
3. *Noise and overfitting*: how do we deal with the adverse effects of noise in learning, in particular by using regularization and validation.

We have observed that when equipped with these foundations, the students, including both practitioners and researchers, can then move on to one technique or another with ease and are able to appreciate the larger framework within which those techniques fit. We use this core in our courses as the immutable foundation for a short course in introductory machine learning to both practitioners and researchers; we then *complement* this foundation with specific techniques and algorithms that we wish to emphasize within a particular context, and these specific techniques may change from year to year.

In this position paper, we briefly introduce these three key concepts in the foundation, and present a simple case of how to position new techniques within this foundation. We welcome discussions on what should be added or removed from the foundations. Our hope is to highlight the foundation-based approach in teaching machine learning, an approach that seems to be getting neglected in current teaching trends that mostly focus on presenting as many flashy techniques as possible.

2. Learnability, Approximation and Generalization

– what we see is not what we get

For beginners in machine learning, it is all too often that they would throw data at a learning system and be surprised when they come to deploy the learned system on test examples. The first principle that should be instilled early on is that what we see in-sample is not always what we get out-of-sample. We need to emphasize that the out-of-sample performance is what we care about, and this leads to a natural question: how can we claim that learning (getting a low out-of-sample performance) is feasible, while only being able to observe in-sample? And so, we arrive at the logical reformulation of the learning task into two steps.

1. Make sure we have good in-sample performance;
2. Make sure in-sample performance reflects out-of-sample performance.

There are several advantages to emphasizing the two step approach. First, it crystallizes the tradeoff between *approximation* (the ability to ensure step 1) and *generalization* (the ability to ensure step 2). Second, it clearly separates learning into the process we can observe (step 1) and the process we cannot observe (step 2). Both steps are *necessary*, but there is an asymmetry: since step 2 cannot be observed, it must be *inferred* in a probabilistic manner. There is a deep consequence of not being able to observe step 2, namely that via powerful and abstract methods, we *must* ensure it, lest it be our unseen Achilles heel. This enables us to characterize successful learning according to what is happening in step 1, in-sample, *modulo that we have ensured step 2*. If we managed to get the in-sample performance we want, then we are done, but if not, we failed, *but at least we will know we failed*.

Contrast this with the typical approach of the beginner: throw a complex model at the data and try to knock down the in-sample error to get good in-sample performance; then hope that things work out luckily during deployment on test examples. This is not the mode we advocate. The mode we advocate is *ensure* step 2 and the try our best with the in-sample performance.

> *We cannot guarantee learning. We can 'guarantee' no disasters. That is, after we learn we will either declare success or failure, and in both cases we will be right.*

There are different levels of mathematical rigor at which this message can be told, but we feel that the essence of the message is a must. Armed with this insight, the enlightened student will rarely falter. It motivates them to *understand* when step 2 can be ensured and the need for the abstract side to learning. It motivates the practical side too – knocking down the in-sample performance.

In our teaching experience, we find that students from diverse backgrounds are highly motivated after encountering the learnability issue, mostly because the issue makes machine learning a non-trivial and fascinating field to them.

3. Linear Models

– simple works well in practice

It is just a dogma that the beginner when faced with the data must have a go at it with the latest and greatest. Armed with an understanding of learnability, approximation and generalization, students instantly see how the linear model quantitatively ensures step 2 (generalization) by suitably controlling the dimension. So all that remains is step 1, and efficient algorithms exist to knock down the error. If that does not work, careful use of the nonlinear feature transform usually solves the problem. In fact, recent advances in large-scale learning suggest that linear models are more successful than current dogma gives them credit for (Yuan et al., 2012).

So the recipe is simple. Try the linear model *first*. It is simple and often effective and not much can go wrong. If the in-sample performance of a linear model is good, we confidently assert success. We always have the option, later, to go with a more complicated model, but with care. And, the linear model is typically the building block for the most popular complex models anyway.

In our teaching experience, we introduce the linear models within three related contexts of classification, regression, and probability estimation (logistic regression). Each context comes with learning algorithms and good generalization. The algorithms illustrate different optimization methods: combinatorial, analytic, and iterative, and so a lot of territory can be covered early by the instructor. Students coming from diverse backgrounds not only get the big picture, but also the finer details in a concrete setting.

4. Noise and Overfitting

– complex situations call for simpler models

Overfitting, the troubling phenomenon where pushing down the training error no longer indicates that we will get a decent test error, is arguably one of the most common traps for beginners. Intuitively, the ubiquitous stochastic noise that corrupts any finite data set can even lead the linear model astray. We should also not lose sight of the other major cause of overfitting: *deterministic noise*, the extent to which the target function cannot be modeled by our learning model (Abu-Mostafa et al., 2012).

> *Whenever there is noise, extra precautions are called for. This goes for stochastic as well as deterministic noise.*

It is counterintuitive to a student that if the target function gets *more complex* we should choose a *simpler learning model*. But, that is why proper understanding of machine learning matters for all students, regardless of what domain their application is in; what separates the amateur from the professional is the ability to deal with overfitting, in particular how to deal with both types of noise.

We recommend a substantive treatment of two basic tools, regularization and validation. An understanding of overfitting sets a perfect stage to introduce regularization and validation. Regularization constrains the model complexity to prevent overfitting, and validation allows one to certify whether a model is good or bad.

These three core concepts: learnability, simple models and overfitting continue to be crucial when we move to the complex models. Each of those models have their own ways of dealing with these issues and so the background set by this core is the platform from which to learn these more complex models (such as neural networks or support vector machines).

5. Support Vector Machines?

We cannot avoid the more complex methods but the advantage of having given the core is that students are fully equipped to study any method independently. For example, the neural network is just a cascade of linear models. Ensemble methods are typically combinations of linear models voted together in some principled way. Support Vector Machines (SVM; Vapnik, 1998), which we now focus on, are linear models with a 'robust' formulation that can be solved with quadratic programming; and, the nonlinear transform can be incorporated efficiently with SVM using the kernel trick.

One reason to choose SVM (for classification) is because it is currently the most popular. In reality, most models are good, it's just that SVM requires the least amount of 'expertise' to use effectively. But given an understanding on the foundations, the student is equipped to understand the *why* and the *how*. Again the discussion can be taken to a mathematical level of the instructor's choice, but the concepts will remain firm.

How does SVM accomplish the two steps required for learnability? Generalization is handled because the model is *simple* - it is a linear model with an ingredient of margin control to ensure the generalization performance (step 2) via theoretical results.

What about getting good in-sample performance? First, there is an efficient algorithm, based on quadratic programming. And if that fails, we can exploit the nonlinear transform efficiently using a kernel. What if our data is too large for general quadratic programming tools? The many other tools that are available for linear models, such as stochastic gradient descent (SGD) commonly used for linear logistic regression, can be taken with linear SVM to deal with scalability.

What about overfitting? We can efficiently estimate the leave-one-out error and hence validate (certify) the model. In addition, the primal form of SVM asks to minimize the norm of the weight vector subject to fitting the data - well, that is a form of regularization.

So we see why SVM is popular among the diverse machine learning crowd – lots of things are taken care of automatically. That doesn't mean that these things cannot be done with neural networks. And there lies the effectiveness of the foundation-based approach. SVM is not some magic box. It is just one way of realizing the three core concepts.

6. Conclusion

We discussed a proposed three-step core as a foundation for machine learning in both practice and theory. The three concepts cover the philosophical understanding (when learning is possible), the algorithmic modeling (how to learn with simple models), and the practical tuning (how to combat overfitting). We argue that the foundation can solidify the understanding of students coming from diverse backgrounds, and can be carried through to any technique, simple or complex. Ironically, we are not proposing that we alter the teaching to accommodate more diverse audiences. Quite the opposite, the foundations-based approach is all the more important for such audiences to ensure that they use machine learning correctly.

References

Abu-Mostafa, Yaser S., Magdon-Ismail, Malik, and Lin, Hsuan-Tien. *Learning from Data: A Short Course*. `AMLBook.com`, 2012.

Vapnik, Vladimir N. *Statistical Learning Theory*. Wiley, New York, NY, 1998.

Yuan, Guo-Xun, Ho, Chia-Hua, and Lin, Chih-Jen. Recent advances of large-scale linear classification. *Proceedings of IEEE*, 2012. doi: 10.1109/JPROC.2012.2188013. to appear.