Hierarchial Multiclass Classification Works Better Than Direct Classification: An Explanation of the Empirical Fact

Julio Urenda[1,2], Nancy Avila[3],
Nelly Gordillo[4], and Vladik Kreinovich[2]
[1]Department of Mathematical Sciences
[2]Department of Computer Science
[3]Biomedical Engineering Program,
Department of Metallurgical, Materials and Biomedical Engineering
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
nsavila@utep.edu, jcurenda@utep.edu, vladik@utep.edu
[4]Department of Electrical and Computer Engineering
Universidad Autónoma de Ciudad Juárez
Ave. del Charro No. 450 Norte
Ciudad Juárez, Chihuahua, México 32310
nelly.gordillo@uacj.mx

Abstract

Machine learning techniques have been very efficient in many applications, in particular, when learning to classify a given object to one of the given classes. Such classification problems are ubiquitous: e.g., in medicine, such a classification corresponds to diagnosing a disease, and the resulting tools help medical doctors come up with the correct diagnosis. There are many possible ways to set up the corresponding neural network (or another machine learning technique). A direct way is to design a single neural network with as many outputs as there are classes – so that for each class i, the system would generate a degree of confidence that the given object belongs to this class. Instead of designing a single neural network, we can follow a hierarchical approach corresponding to a natural hierarchy of classes: classes themselves can usually be grouped into a few natural groups, each group can be subdivided into subgroups, etc. So, we set up several networks: the first classifies the object into one of the groups, then another one classifies it into one of the subgroups, etc., until we finally get the desired class. From the computational viewpoint, this hierarchical scheme seems to be too complicated: why do it if we can

use a direct approach? However, surprisingly, in many practical cases, the hierarchical approach works much better. In this paper, we provide a possible explanation for this unexpected phenomenon.

1 Formulation of the Problem

Case study: classification in medicine. One of the most challenging tasks in medicine is diagnostics. The main reason why this task is difficult is that usually, several different diseases have the same set of symptoms, and separating them is therefore not easy. To help medical doctors, researchers have been designing automatic software tools that provide a preliminary classification of patients into groups corresponding to different diseases. Of course, in their present form, these tools do not provide a perfect diagnostics, they cannot replace the medical doctors. However, these tools can be (and are) of help to medical doctors. They are especially useful for medical doctors who are just starting their medical careers and who therefore do not yet have the experience that comes from observing and treating hundreds and thousands of patients.

In particular, one of such tools has been developed by Nancy Avila – one of the authors of this paper – for classifying children's lung disorders [1].

How classification is usually obtained: general idea. One of the most widely used technique to get the desired classification is to use machine learning:

- we first train the algorithm on examples for which the classification is known;
- once the algorithm has been trained, we apply the resulting trained algorithm to new inputs and thus, produce the class that – according to this tool – contains this input.

At present, the most efficient machine learning tool is neural networks (see, e.g., [2, 3]), but other machine learning tools have been (and are) used to solve classification problems.

Multi-class classification: details. In many classification problems, we need to classify objects into several different classes. Let n denote the number of such classes.

In an ideal situation when, based on the evidence, it is absolutely clear that the object belongs to one of the classes, the system should produce this class. In many practical situations, however, even with all the available evidence, we are often not 100% sure what is the most appropriate class. In this case, we would like the system not only to produce one class, but to describe all possible classes – and for each possible class, to provide a degree of confidence that the object is in this class. This will definitely help the user to make a final decision.

In other words, to perform an n-class classification, we would like to design a system with n outputs $y_1, \ldots, y_n$, where each y_i describes to what extent the existing evidence supports the conclusion that the given object belongs to the

i-th class. A user would also like to know the most probable class – i.e., in terms of the values y_i, the class i for which the degree of confidence y_i is the largest.

Direct approach. In the direct approach, we set up a neural network (or another machine learning tool) with n outputs, and we train it on the examples in which we know the class i_0, i.e., on the examples in which we have $y_{i_0} = 1$ and $y_i = 0$ for all $i \neq i_0$.

Hierarchical approach. Another alternative is to take into account that neural networks are, after all, attempts to simulate how we humans process data. Thus, when using neural networks, it makes sense to recall how we ourselves perform the corresponding analysis.

From this viewpoint, the direct approach is not how, e.g., medical doctors diagnose a patient – and, in general, not how we classify objects. Neither medical doctors not we humans in general keep in mind all thousands of possible alternatives. Instead, our classification is usually a hierarchical, multi-step one that goes from the general to the particular. For example, if we hear some not-very-loud noise when walking at night on a country road, we first check whether it is a human or an animal or something brought in by the wind. Then, if, e.g., it is an animal, we decide whether it is big (and possible dangerous) or small, if small whether it is a pet or, e.g., a rabbit, etc. The same logical process happens with doctors when diagnosing a disease, they start with general tests (i.e. blood testing) to identify normal or abnormal conditions; if an abnormal condition is observed, further testing (more specific) is performed to diagnose a disease.

In such cases, we reduce the large classification problem to several smaller-size ones, e.g., to several binary (2-class) classifications.

Empirical fact: hierarchical approach works much better. The direct approach is easier to implement: you just need to train one neural network. As a result, this is what people usually start with. Interestingly, this approach often does not work well, while the more difficult-to-implement hierarchical approach leads to much better learning and thus, to a much more accurate classification. This is, e.g., exactly what happened in the medical classification problem analyzed in [1].

How can we explain this empirical fact? In this paper, we provide a possible explanation for this empirical fact.

2 Our Explanation

Analysis of the problem. We are interested in situations in which classification is difficult – otherwise, if it was easy, we would not need to design a complex computer-based tool to help the users.

In our scheme, we classify an object into a class i for which the degree of confident y_i (estimated by the system) is the largest. In these terms, *difficult* means that in many cases, it is difficult to decide which of the values y_i is the

largest – i.e., that there are several classes i with approximately the same value of the degree of confidence.

For the "ideal" values $Y_1, \ldots, Y_n$ (that we would have gotten if we had all the information and no noise) we have $Y_{i_0} > Y_i$ for all $i \neq i_0$. However, because of the noise (and incompleteness of data) the estimates y_i are, in general, different from Y_i.

If the effect of the noise is that one of the values y_i with $i \neq i_0$ is "boosted", i.e., gets a reasonable-size positive estimation error $\Delta y_i \stackrel{\text{def}}{=} y_i - Y_i$, then the boosted value $y_i = Y_i + \Delta y_i$ becomes larger than $y_{i_0} \approx Y_{i_0}$ and thus, we get an incorrect classification.

To analyze whether a direct approach or a hierarchical approach work better, let us estimate, for each of these two approaches, the probability of such a misclassification.

What is the probability of a misclassification: case of the direct approach. Let p be a probability that one of the outputs gets boosted. Then, in the direct approach, we get a misclassification if one of the $n-1$ wrong outputs gets boosted.

Each of these boostings is an independent event, so the probability that none of the $n-1$ boosting occurs can be estimated as a product of $n-1$ probabilities (equal to $1-p$) that each of the boostings will not happen, i.e., as $(1-p)^n$. Thus the probability that a misclassification will happen is equal to 1 minus this probability, i.e., to $1-(1-p)^{n-1}$.

The probability p is usually small – otherwise, the classification would be lousy. For small p, we can expand the above expression in Taylor series in p and ignore quadratic (and higher order) terms in this expansion. As a result, we conclude that the probability of misclassification if approximately equal to $(n-1) \cdot p$.

What is the probability of a misclassification: case of the hierarchical approach. To make an estimation, let us consider the case when on each stage of the hierarchical classification, we have a binary classification – i.e., a classification into two classes. In this case, with one stage, we can classify objects into 2 classes; with 2 binary stages, we can classify them into 4 classes, and, in general, with s binary stages, we can classify objects into 2^s classes. Thus, to get a classification into n classes, we need to select the number of stages s for which $2^s \approx n$, i.e., we need $s \approx \log_2(n)$ stages.

On each stage, we compare two numbers y_i: the number corresponding to the correct group (or subgroup) and the number corresponding to an incorrect one. Here, it is also possible that boosting will resulting in a wrong subgroup. The probability of this, on each of the s stages, is p. If at one of the s stages, we get a wrong subgroup, we get a misclassification. Similarly to the direct case, we can conclude that the probability of misclassification is equal to $1-(1-p)^s \approx s \cdot p \approx \log_2(n) \cdot p$.

If we have 3 classes on each stage, we would get $2\log_3(n) \cdot p$. If we had 4 classes per stage, we would get $3\log_4(n) \cdot p$, etc.

The resulting explanation. In the direct approach, the probability of misclassification is $n \cdot p$, while in the hierarchical approach, this probability if $\log_2(n) \cdot p$. Since for large n, we have $\log_2(n) \ll n$, this shows that indeed the hierarchical classification is much more accurate – exactly as the empirical data shows.

This conclusion does not change if on each stage, we classify into $c \neq 2$ classes: we would get $(c-1) \cdot \log_c(n) \cdot p$ which is still much smaller than $n \cdot p$.

Acknowledgments

This research was partially supported:

- by the National Council of Science and Technology of México (CONACYT) via a doctoral fellowship for Nancy Avila,
- by the Research Program on Migration and Health (PIMSA), a program funded by the University of California, Berkeley, Health Initiative of the Americas (HAI), and
- by the US National Science Foundation via grant HRD-1242122 (Cyber-ShARE Center of Excellence).

References

[1] N. Avila, *Computer-Aided Classification of Impulse Oscillometric Measures of Respiratory Small Airways Function in Children*, PhD Dissertation, Biomedical Engineering Program, University of Texas at El Paso, 2019.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016.